MethylExplorer: buscador de estudios de metilación de ADN

José Luis Esteban González
Máster en Bioinformática y Bioestadística
Trabajo Fin de Máster

Director TFM: Alex Sánchez Pla

24/05/2017

FICHA DEL TRABAJO FINAL

Título del trabajo:	*MethylExplorer: buscador de estudios de metilación de ADN*
Nombre del autor:	*José Luis Esteban González*
Nombre del PRA:	*Alex Sánchez Pla*
Fecha de entrega (mm/aaaa):	05/2017
Titulación::	*Master en Bioinformática y Bioestadística*
Área del Trabajo Final:	*Trabajo Fin de Máster*
Idioma del trabajo:	***Castellano***
Palabras clave	*Metilación, buscador, repositorios*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Los estudios de metilación de ADN generan gran cantidad de datos que se almacenan en distintos repositorios, y en general se publican como datos abiertos. MethylExplorer localiza este tipo de datos en tres de estos repositorios (GEO, ArrayExpress y GDC), y de forma centralizada permite obtener y analizar algunos de ellos. La aplicación está desarrollada con Shiny y R, y se puede desplegar en el servidor Shiny Server, de forma que sea accesible de forma remota.

Abstract (in English, 250 words or less):

DNA methylation studies generate large amounts of data that are stored in different repositories, and are generally published as open data. MethylExplorer locates this type of data in three of these repositories (GEO, ArrayExpress and GDC), and centrally allows to obtain and analyze some of them. The application is developed with Shiny and R, and can be deployed on the Shiny Server, so that it is accessible remotely.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Existen varios mecanismos epigenéticos que regulan la expresión génica en las células: la metilación en el DNA, la modificación de las histonas que interaccionan con dicho DNA y el RNA no codificante, que incluye lncRNA y sncRNA (miRNA, siRNA, piRNA).

La modificación de histonas engloba un conjunto de reacciones químicas que incluye metilaciones y acetilaciones sobre estas proteínas que rodean al DNA, y se habla del código de histonas debido a la complejidad de sus posibilidades de combinación y efectos producidos. De forma general, se puede afirmar que la función principal es regular el estado de la cromatina en sus diferentes estados, de forma que el DNA sea accesible o no (silenciamiento). El RNA no codificante en sus diferentes formas también juega un papel fundamental en la regulación génica como mediador, tanto en su interacción con las histonas como con otras proteínas que intervienen a nivel transcripcional.

En tercer lugar, la metilación del DNA actúa de forma directa añadiendo un grupo metilo, que en células de mamíferos se realiza en la posición C5 (molécula de citosina); la forma abreviada 5mC indica que se encuentra metilada. A diferencia de los mecanismos anteriores, y por los estudios realizados hasta la fecha, es el único heredable.

El proceso de metilación lo realiza la família de enzimas DNMTs, en las que se distinguen las que realizan metilación de novo (DNMT3A, DNMT3B entre otras), y las que mantienen las señales de metilación (DNMT1).

La metilación del DNA está asociada al silenciamiento de genes, y durante las fases de desarrollo los patrones de metilación se modifican. En fases tempranas por ejemplo, se produce borran y posteriormente se restablecen las señales de metilación en todo el genoma excepto en las islas CpG, o agrupaciones de secuencias CpG de hasta unos pocos kb de longitud y que pueden suponer aproximadamente un 2% del genoma. Estas islas CpG se suelen localizar en los promotores de los distintos genes a los que afectan, suponiendo una gran concentración del total de la composición de bases de dicho promotor.

Además de las consecuencias asociadas a la función de la metilación a nivel celular, hay que tener en cuenta otros efectos, como la metilación en regiones con SNPs. Por tanto, son otro factor en los análisis realizados sobre el estado de la metilación en distintos lugares del genoma, como en metilación diferencial. También se busca identificar otras regiones como las CpG shores, que pueden tener relación con las islas CpG y están a una distancia de hasta 2 kb de éstas, o las CpG shelves a una distancia de hasta 4 kb.

Cada metodología o tipo de estudiio relacionado con la metilación está enfocado en unos objetivos específicos, de ahí que se usen diferentes enfoques o estrategias; en ocasiones se busca obtener un valor de metilación global del genoma estudiado, en concreto, hipo e hipermetilación, ésta última asociada a envejecimiento, algunas enfermedades y desarrollo de tumores. En este tipo de metodologías estarían HPLC o LUMA entre otras, como se describe en [1]. Estos métodos suelen tener un menor coste y se suelen emplear como base en un estudio epigenético.

Otros estudios están más enfocados en regiones concretas del DNA o en todo el genoma, pero con mayor resolución (base DNA). En esta categoría se incluyen WGBS, RRBS, microarrays de metilación, MeDIP o Tab-Seq entre otras. Son técnicas que tienen diferentes ventajas e inconvenientes, y se seleccionan en función de los objetivos y presupuestos asignados a los diferentes estudios. En general, se basan en un tratamiento previo con bisulfito sódico para distinguir las citosinas metiladas de las no metiladas, de forma que el bisulfito facilita la deaminación de la citosina no metilada que pasa a uracilo, y se lee como timina, a diferencia de la metilada o 5mC que resiste a la conversión. Posteriormente, se aplican métodos de enriquecimiento y PCR para amplificar las muestras antes de proceder a la lectura por la plataforma correspondiente.

Algunas consideraciones a tener en cuenta son la preparación de las muestras, el coste o la cobertura. De forma general, las técnicas de secuenciado tienen una mayor cobertura del genoma y generan por tanto, una mayor cantidad de datos a analizar. Además, necesitan un preprocesamiento con mayor coste computacional sobre todo en el alineamiento de secuencias, y pueden localizar regiones CpG nuevas. Las técnicas basadas en microarrays se basan en un proceso de hibridación con las ventajas e inconvenientes que genera, tienen una cobertura menor que las de secuenciado aunque considerable (95% de las islas Cpg y 99% de los promotores RefSeq con la plataforma Illumina MethylationEPIC, 850K). Un diseño adecuado del estudio permite estimar la variabilidad de las muestras a nivel biológico, e influye en la interpretación de los análisis realizados.

Una vez realizados los diferentes ensayos de los que consta el estudio, los datos obtenidos se tratan mediante procesos de preprocesamiento específicos de cada plataforma, en los que se buscan eliminar los sesgos introducidos en el estudio (proceso de hibridación, preparación de muestras, etc) para obtener un valor que resuma el estado de metilación en las distintas sondas o posiciones. En los estudios de metilación con microarrays el valor que resume el nivel de metilación es un ratio entre las sondas hibridadas y el total (metiladas o no). En [2] se analiza el valor beta y su análogo en escala logarítmica M value en el que se discuten las ventajas de cada uno. La ventaja del valor beta o beta value radica en que es fácilmente interpretable a nivel biológico, ya que se encuentra en una escala entre 0 y 1, mientras que M value tiene

una menor heterocedasticidad en los análisis de metilación diferencial, y ofrece por tanto, mejores parámetros a nivel estadístico.

La fase de preprocesamiento de los datos RAW es muy específica sobre todo en los microarrays, debido a la naturaleza de las diferentes sondas usadas. Se puede ilustrar con el diseño de las sondas de la plataforma Illumina HumanMethylation450K.

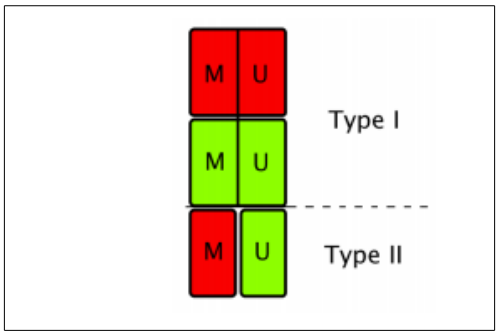


Ilustración 1: Esquema del diseño de sondas usado en la plataforma Illumina HumanMethylation450

La ilustración muestra dos tipos de sondas usadas por la plataforma, en las que las sondas de tipo 1 miden las dos posibles señales en el mismo canal o color, mientras que en las de tipo 2 se usa un canal para cada señal. El preprocesamiento incluye la verificación de la calidad de la señal, así como procesos para discriminar el background provocado por la naturaleza de la plataforma. El análisis de los perfiles de metilación obtenidos incluye la localización de DMR entre las diferentes muestras que componen el estudio, así como niveles de metilación globales entre los grupos que se hayan diseñado.

En cuanto al software opensource existente relacionado con el análisis de datos masivos producidos con este tipo de técnicas existen varios paquetes en el repositorio Bioconductor como Bsmooth, methylKit, methylSig, methylPipe. También existe software específico comercial de los distintos proveedores de plataformas, como Genome Studio en el caso de Illumina.

Por otro lado, existen una variedad de repositorios que publican los resultados de diferentes estudios de metilación de DNA, como MethBase, MethylomeDB, DiseaseMeth, y otros menos específicos y bastante extendidos como GEO, ArrayExpress o GDC; éste último engloba diferentes repositorios. La aportación realizada con el desarrollo de la aplicación MethylExplorer es contar con una herramienta capaz de buscar información de metilación de DNA en los repositorios GEO, ArrayExpress y GDC, en los que conviven con otros tipos de estudios. También se busca facilitar la extracción de los datos involucrados, así como disponer de herramientas de análisis que permita una visualización de los resultados.

Otra estrategia para la exploración de datos, y que aporta un valor añadido es el desarrollo centrado en el análisis de muestras, que pueden provenir de un mismo o diferentes estudios. Al comparar datos de diferentes estudios el usuario es responsable en la selección de muestras que introducirá una variabilidad biológica adicional cuando se traten distintos estudios, y debería tenerse en cuenta en la interpretación posterior de los análisis realizados.

1.2 Objetivos del Trabajo

Los objetivos que se describen a continuación resumen la funcionalidad de la aplicación en sus diferentes apartados:

- Localizar estudios de metilación de DNA en los repositorios GEO, ArrayExpress y GDC, en base a unos criterios de unos criterios de selección, implementados mediante filtros.

- Personalizar las búsquedas mediante diferentes parámetros de usuario.

- Proporcionar información de las distintas muestras involucradas, para facilitar una selección de muestras eficiente.

- Realizar un análisis de las muestras seleccionadas, garantizando que proceden de las mismas plataformas.

- Proporcionar información del tipo de datos (raw/procesados) que se dispone de las diferentes muestras, así como la opción de descarga de los datos RAW de las muestras seleccionadas para su posible procesamiento con otras aplicaciones.

- Visualizar los datos tanto a nivel global, como en su contexto genómico.

- Mostrar estadísticas de las distintas consultas para tener una visión global de los resultados, y una pantalla de ayuda rápida en el uso de la aplicación.

Objetivos más se definen en función de las distintas estrategias de obtención de datos de los distintos repositorios, y se exponen en el apartado de desarrollo de la aplicación.

1.3 Enfoque y metodología

Se plantearon diferentes estrategias en el inicio del proyecto, para lograr los objetivos marcados. De entre los descartes, una primera aproximación fue la realización de una librería o paquete en R que centralizarla los procesos de obtención de datos y análisis mediante distintas funciones.

Uno de los objetivos que está implícito en los objetivos que se han definido es dotar de la mayor interactividad al usuario, por tanto la estrategia de desarrollo usada es la realización de una aplicación web, con las tecnologías Shiny+R en la parte del servidor, y que sea accesible online mediante HTTP.

Esta estrategia de desarrollo permite integrar el proceso completo de obtención de estudios, selección de muestras y análisis de datos procesados, y también facilita una futura actualización mediante la inclusión de módulos adicionales, tanto de análisis como de acceso a otros repositorios.

El ciclo de vida de desarrollo seguido se ha basado en el prototipado modular, con las fases clásicas de diseño, implementación y pruebas en los diferentes módulos y de forma global, y pruebas de integración y despliegue posterior.

Otro de las cuestiones a tener en cuenta, inherentes a la tecnología Shiny es el uso de objetos reactivos que se actualizan de forma dinámica con determinadas acciones del usuario. Esto facilita el diseño modular con la construcción de módulos autocontenidos, y añade algo de dificultad al integrar los módulos, haciendo necesario el aislamiento de código en algunos casos.

La reutilización de código está presente a lo largo del desarrollo de todo el proyecto, mediante el uso de distintos paquetes de acceso o análisis a los diferentes datos.

1.4 Planificación del Trabajo

Las distintas fases en las que se ha realizado el trabajo se pueden resumir en el diagrama de Gantt que se muestra a continuación.

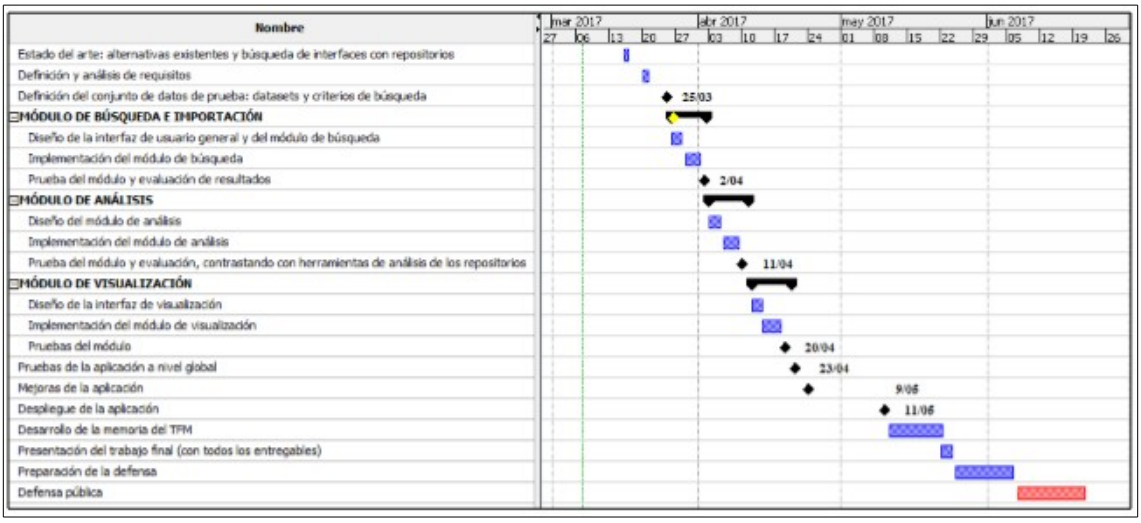


Ilustración 2: Diagrama de Gantt del trabajo

Las distintas tareas que se definieron en el plan de trabajo se han cumplido con éxito, a excepción de la fase de mejoras de aplicación que ha servido para solucionar algunos problemas de configuración en el despliegue. También se han realizado mejoras que se describen en el capítulo correspondiente. El *listado de tareas* definitivo es el que se describe, sin incluir las fases de elaboración de este documento y posteriores tareas.

- Estado del arte: alternativas existentes y búsqueda de interfaces con repositorios
  - Definición y análisis de requisitos
  - Definición del conjunto de datos de prueba: datasets y filtros
- MÓDULO DE BÚSQUEDA E IMPORTACIÓN
  - Diseño de la interfaz de usuario general y del módulo de importación.
  - Implementación del módulo de búsqueda
  - Prueba del módulo y evaluación de resultados
- MÓDULO DE ANÁLISIS
  - Diseño del módulo de análisis
  - Implementación del módulo de análisis
  - Prueba del módulo y evaluación
- MÓDULO DE VISUALIZACIÓN
  - Diseño de la interfaz de visualización
  - Implementación del módulo de visualización
  - Pruebas del módulo
- Pruebas de la aplicación a nivel global
- Mejoras de la aplicación

- Despliegue de la aplicación

Los hitos que se han seguido para evaluar los distintos objetivos parciales han sido los siguientes:

- Definición de requisitos, evaluación y definición de datos de prueba
- En la finalización del módulo de importación con cada uno de los filtros desarrollados (uno por repositorio).
- Al finalizar el módulo de análisis.
- Al finalizar el módulo de visualización de resultados.
- Al finalizar las pruebas de coherencia de la aplicación, verificando la correcta conexión de los diferentes módulos.
- Al finalizar las mejoras desarrolladas.
- Al finalizar el despliegue de la aplicación.

La duración en el desarrollo del trabajo ha sido de unos dos meses aproximadamente, contando hasta la fecha de entrega desde el comienzo el 16/03/2017 hasta el 24/05/2017. Se muestra el diagrama de tiempos:

Nombre	Inicio	Terminado
Estado del arte: alternativas existentes y búsqueda de interfaces con repositorios	16/03/17 8:00	17/03/17 8:00
Definición y análisis de requisitos	20/03/17 8:00	21/03/17 16:00
Definición del conjunto de datos de prueba: datasets y criterios de búsqueda	25/03/17 8:00	25/03/17 16:00
MÓDULO DE BÚSQUEDA E IMPORTACIÓN	**26/03/17 8:00**	**9/04/17 17:00**
Diseño de la interfaz de usuario general y del módulo de búsqueda	26/03/17 8:00	28/03/17 17:00
Implementación del módulo de búsqueda	29/03/17 8:00	1/04/17 17:00
Prueba del módulo y evaluación de resultados	2/04/17 8:00	9/04/17 17:00
MÓDULO DE ANÁLISIS	**10/04/17 8:00**	**17/04/17 17:00**
Diseño del módulo de análisis	10/04/17 8:00	12/04/17 17:00
Implementación del módulo de análisis	12/04/17 8:00	15/04/17 17:00
Prueba del módulo y evaluación, contrastando con herramientas de análisis de los repositorios	16/04/17 8:00	17/04/17 17:00
MÓDULO DE VISUALIZACIÓN	**18/04/17 8:00**	**27/04/17 17:00**
Diseño de la interfaz de visualización	18/04/17 8:00	20/04/17 17:00
Implementación del módulo de visualización	21/04/17 8:00	25/04/17 17:00
Pruebas del módulo	26/04/17 8:00	27/04/17 17:00
Pruebas de la aplicación a nivel global	28/04/17 8:00	30/04/17 17:00
Mejoras de la aplicación	1/05/17 8:00	9/05/17 17:00
Despliegue de la aplicación	10/05/17 8:00	11/05/17 17:00
Desarrollo de la memoria del TFM	11/05/17 8:00	22/05/17 17:00
Presentación del trabajo final (con todos los entregables)	22/05/17 8:00	24/05/17 17:00
Preparación de la defensa	25/05/17 8:00	6/06/17 17:00
Defensa pública	7/06/17 8:00	21/06/17 17:00

Ilustración 3: Diagrama de tiempos de cada tarea

Los medios necesarios para la realización del proyecto han sido los siguientes:

Hardware	Portátil HP Pavilion dv6 con procesador Intel Core i7-2670QM y 4 Gb de memoria RAM
Software	R compilación 3.4.0
	R Studio 1.0.143
	Ubuntu Server 17.04 virtualizado con VirtualBox 5.1.22
	Shiny Server Open Source
Conexión ADSL	Propiedad de Biblioteca Pública de Guadalajara

1.5 Breve sumario de productos obtenidos

Los entregables finales que ya se enumeraron en el plan de trabajo son los que se describen a continuación:

1. Plan de trabajo que incluye la estimación temporal, y los objetivos definidos en su fase inicial.

2. Memoria que representa el presente documento, y donde se detallan las distinfas fases, funcionalidades, limitaciones y conclusiones del trabajo realizado.

3. Código fuente de la aplicación.

4. Vídeo de presentación del trabajo y diapositivas que lo integran.

5. Informe de autoevaluación, en el que se comentan distintas cuestiones adicionales que han marcado el desarrollo del proyecto, y que no están descritas en la memoria.

1.6 Breve descripción de los otros capítulos de la memoria

Contenidos de cada capítulo contextualizados en el marco de desarrollo:

En el *capítulo 2*, se detallan todas las cuestiones en relación a las distintas fases de la aplicación, y se ha desglosado en las siguientes partes:

El *apartado 2.1* se ocupa de la definición de requisitos, selección de datos de prueba, y su análisis.

El *apartado 2.2* explica las estrategias de diseño de los diferentes interfaces gráficos, y las capas backend de almacenamiento y uso de los datos en los distintos módulos. Se puede considerar el capitulo principal para la comprensión del funcionamiento interno de la aplicación.

El *apartado 2.3* contiene la explicación de cómo se ha gestionado la persistencia, así como de los objetos que se cargan en memoria.

El *apartado 2.4* puede considerarse un resumen de las distintas evaluaciones realizadas en las pruebas con los diferentes módulos, y cómo éstas han influido en las mejoras de la aplicación. Además, se incluye información de los distintos mensajes implementados para informar al usuario y de la gestión de errores en general.

El *apartado 2.5* es una guía para la configuración del servidor en el que estará alojada la aplicación.

El *último apartado 2.5* indica las mejoras realizadas respecto a los diseños iniciales. Es importante destacar que sólo se entrega la versión final de la aplicación mejorada, de ahí la inclusión de este apartado.

En el *capítulo 3* se incluyen las limitaciones de la aplicación, así como futuras ampliaciones y mejoras que no han podido llevarse a cabo.

El *capítulo 4* es una reflexión global del producto obtenido, teniendo en cuenta las limitaciones del capítulo anterior.

2. Desarrollo de la aplicación

En este apartado se describen todas las cuestiones relacionadas con las distintas fases del desarrollo del producto, desde la definición y análisis de requisitos inicial hasta la fase final de despliegue en el servidor. Se usan diferentes capturas de pantalla sobre todo para clarificar cuestiones en el diseño.

2.1 Definición de requisitos y análisis

- Recuperación de estudios con características definibles por el usuario. Con esto se ha entendido que mediante una interfaz de usuario personalizable se puedan seleccionar los diferentes criterios propios de los estudios almacenados en los repositorios GEO, TCGA.

- Exploración mediante unos procedimientos básicos, como localización de locus diferencialmente metilados. El resultado del análisis es por tanto, una listado de locus en un contexto genómico con sus coordenadas correspondientes.

- Comparación de resultados entre estudios. Este requisito ha sido determinante para realizar un diseño de la aplicación orientado amuestras, de forma que no se limite a importar y analizar estudios completos sino que permita seleccionar subconjuntos de datos.

- Visualización de los resultados. Esta es una de las partes que ha quedado menos cubierta en el trabajo, debido sobre todo a problemas de rendimiento en los recursos utilizados. La visualización debería ser capaz de representar de forma visual los resultados de los análisis, de forma que a partir de unas coordenadas genómicas se pueda observar con elementos de referencia como genes o tránscritos, el perfil de metilación o los DMRs detectados.

Este ha sido el punto de partida del trabajo, y el análisis de estos requisitos se ha completado con las siguientes cuestiones:

- El usuario debería ser capaz de extraer muestras concretas de determinados estudios en los repositorios definidos, al que se ha añadido el repositorio ArrayExpress, ya que en los últimos años ha elevado considerablemente el número de estudios propios almacenados. Esta idea de estudios propios es importante ya que este repositorio también recupera datos de GEO, y los adapta a su nomenclatura. Por tanto, se pueden identificar de forma immediata (E-GEOD-xxxx).

- Debería mostrarse unas estadísticas básicas de los estudios recuperados para que el usuario tenga una visión global de las búsquedas realizadas por la aplicación.

- Dada la gran diversidad de plataformas y estudios, habría que diseñar un sistema para poder seleccionar muestras de estudios analizadas con determinadas plataformas, y también poder discriminar si los datos son procesados o en bruto.

2.2 Diseño e interfaces de los diferentes módulos

La aplicación se ha diseñado para una interfaz tipo *shinydashboard,* en la que se han buscado una proporcionalidad en el diseño, usando elementos de distintas partes de la pantalla. Se pueden distinguir por tanto tres elementos básicos en esta aplicación, y que serían extrapolables a cualquier aplicación desarrollada con este sistema:

Header o parte superior: este elemento se ha usado para diseñar un sistema de notificaciones, sobre todo para identificar si las muestras seleccionadas son equivalentes o no, y para indicar al usuario los posibles problemas ante el intento de obtención de datos RAW o procesados.

Sidebar o barra lateral: está situada en la parte izquierda y alberga las diferentes pantallas de las que consta la aplicación. Se han seleccionado más opciones en el menú que módulos de los que se han definido en los requisitos para no dar una sensación de sobrecarga de datos. Las distintas pantallas, que corresponden a opciones de este menú son:

Seleccionar estudios: permite seleccionar en una tabla de resultados los estudios que se pretenden analizar, en los que cada uno puede pertenecer a un repositorio diferente y características diferentes.

Datos del estudio: permite seleccionar las distintas muestras de cada estudio.

Analizar muestras: permite realizar unos análisis básicos sobre los datos cargados de las muestras seleccionadas

Visualizar resultados: es la pantalla de las distintas gráficas, basadas en los datos obtenidos cargados o de los análisis.

Configuración: permite configurar los filtros de búsqueda de la pantalla de selección inicial, de forma que cada sesión en la que se ejecuta la aplicación pueden tener distintos parámetros.

Estadísticas: muestra con unas gráficas elementales datos de los estudios recuperados en la tabla de resultados. Para el repositorio GDC también se muestran estadísticas específicas de dicho repositorio, que permiten ver entre otras cuestiones, la proporción de muestras procedentes de plataformas de metilación en relación a otro tipo de plataformas (expresión, RNA-seq, etc).

Ayuda: es una guía básica mediante captura de pantalla del funcionamiento de la aplicación. También se ofrece una opción de consulta para buscar plataformas relacionadas con el organismo cargado en los filtros (consulta realizada al repositorio GEO con su nomenclatura correspondiente).

Además, en la parte superior del menú se ha diseñado un campo de introducción de texto, que permite buscar estudios de acuerdo a unas palabras clave definidas.

Parte **central** de la aplicación: constituye el resto de la pantalla, y es aquí donde se realiza el grueso de las acciones del usuario, con excepción de la selección de pantalla.

Una vez revisada la estructura de la aplicación se pasa a explicar los diferentes módulos en los que se ha desglosado el proyecto y que se basan en los requisitos definidos anteriormente.

Módulo de **búsqueda**: es el encargado de mostrar los filtros y resultados de las consultas, y también de seleccionar las muestras de cada estudio. Está formado por tanto, por dos interfaces diferentes en la aplicación.

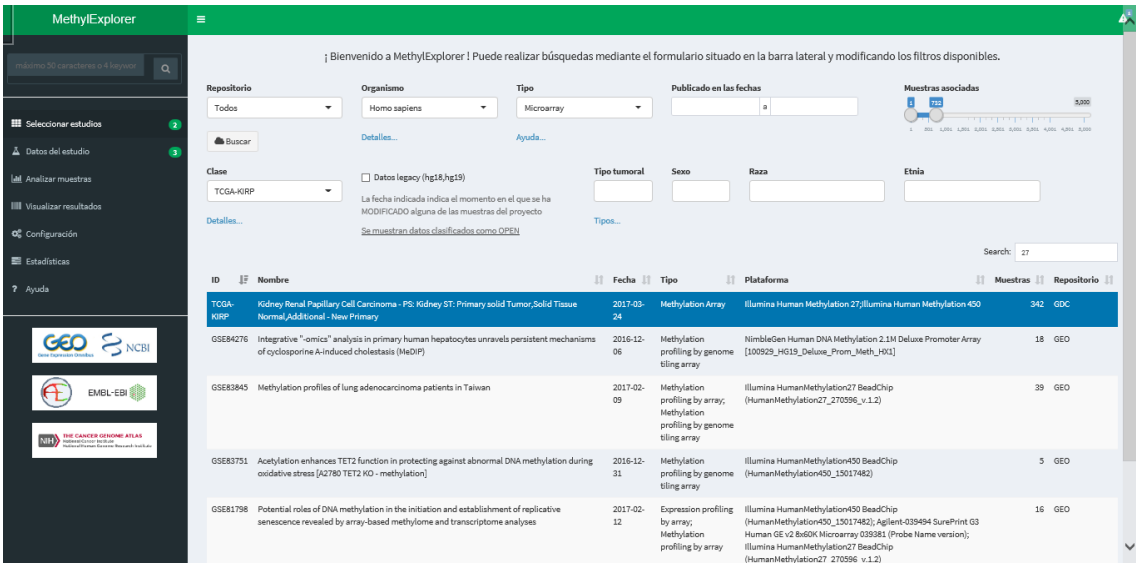


Ilustración 4: Interfaz de consultas de estudios

A continuación se explican los diferentes elementos de diseño, así como su funcionalidad:

Campo de introducción de texto (sidebar): permite introducir hasta un máximo de 4 palabras clave o 50 caracteres en total. Esto se ha diseñado de esta forma entre otras cuestiones por seguridad, para evitar acciones no deseadas en las consultas como SQLi.

Repositorio: con la lista de repositorios a los que se consulta. Las opciones posibles son GEO, ArrayExpress, GDC y Todos, ésta última opción permite lanzar consultas a los tres repositorios con los criterios de búsqueda que se hayan definido. En este aspecto, hay que tener en cuenta que hay filtros que no afectan a repositorios específicos como puede ser el caso de organismo que no afectaría a GDC.

*Organismo: s*e han tomado como referencia los organismos almacenados en el repositorio GEO, que son los que se cargan en el campo de selección. Un aspecto a tener en cuenta es que si se realizan búsquedas sobre GDC y está habilitada la búsqueda por organismo, la selección se actualiza de forma automática a 'Homo Sapiens', ya que este repositorio sólo analiza muestras procedentes de tejido humano.

Plataforma: permite seleccionar una plataforma específica de consulta, basada en la nomenclatura GEO de plataformas, con el prefijo GPLxxxx. Es importante tener en cuenta que la conversión para buscar en el repositorio GEO es bastante sencilla pasando de GPLxxxx a A-GEOD-xxxx.

En este punto, y ante posibles conflictos se ha diseñado un sistema excluyente de criterios entre plataforma y organismo, ya que ambos conceptos pueden considerarse análogos; una plataforma concreta lleva implícito un organismo concreto, pero este sistema permite al usuario usar el que prefiera en función de sus prioridades; el parámetro es configurable desde las opciones. Esto se ha diseñado de esta forma, porque puede haber casos en los que interese recuperar estudios de una plataforma concreta, o de un organismo sin importar la plataforma; un ejemplo sería una consulta con el organismo 'Homo sapiens' en el que se devolverán estudios con plataformas de metilación con dicho orgnismo (Illumina HumanMethylation 27K/450K/EPIC,etc).

También se ha habilitado un enlace denominado 'Detalles...', donde se despliega información almacenada en GEO con la plataforma seleccionada.

Tipo: permite lanzar consultas que busquen estudios con datos de metilación basados en la hibridación de microarrays o basados en secuenciado. Es lo que en algunos repositorios se denomina la estrategia del estudio.

Se basan en las categorías que se ha asignado a los estudios en cada repositorio:

```
catGEO <- c('Methylation profiling by SNP array','Methylation profiling by
       array','Methylation profiling by genome tiling array','Methylation
profiling by high throughput sequencing')

catAE <- c('Methylation profiling by array','Methylation profiling by high
throughput sequencing','Bisulfite-seq','MeDIP-seq')

catGDCDC <- c('DNA Methylation','DNA methylation')
catGDCTy <- c('Methylation Beta Value','Methylation beta value','Bisulfite
sequence alignment','Methylation percentage')
catGDCES <- c('Methylation Array','Methylation array','Bisulfite-Seq')
```

Se puede observar en el fragmento de código, cómo cada repositorio define distintas categorías para cada tipo de estudio de metilación. La

aplicación mapea estas categorías a un sistema de marcado de cada estudio individual, y que indica las siguientes posibilidades:

- **A**: Estudio realizado sólo con plataformas basadas en microarrays
- **S**: Estudio realizado sólo con plataformas basadas en secuenciación (high throughput sequencing, MeDIP-seq, Bisulfite-seq)
- **AS**: Estudio en el que hay involucradas muestras obtenidas a partir de plataformas basadas en arrays y muestras obtenidas en plataformas de secuenciación.

Una vez mapeadas a estos tipos, los resultados de las búsquedas mostrados pueden ser (valores posibles de Tipo):

- Microarray: estudios marcados como A o AS
- Secuenciado: estudios marcados como S o AS
- Todos: estudios marcados como A, S o AS

Se ha incluido una ventana guía para mostrar los tipos de estudio que se consultan según la opción seleccionada.

Fecha de publicación: permite establecer el rango de fechas de publicación del estudio. Para estudios GEO se ha definido una opción adicional que permite buscar en las fechas de registro de la solicitud (submission).

Muestras asociadas: mediante un objeto tipo slider se puede definir el rango en el número de muestras que forman parte del estudio. Hay que tener en cuenta que este número se refiere al total de muestras del estudio y no sólo a las muestras que contienen datos de metilación. Esto es importante en estudios con varias plataformas.

A partir de este punto, los siguientes elementos de consulta aplican al repositorio GDC, y se han definido porque cada proyecto almacenado en este repositorio se considera un estudio. La cuestión es que el número de muestras de algunos proyectos es elevado, y puede sobrepasar las 1000 muestras en algunos proyectos como TCGA-BRCA. Otra función de estos filtros secundarios es que en la pantalla de datos del estudio en la que se seleccionan las muestras aparezcan las muestras que cumplen los criterios definidos en este campo. En la captura relativa a la siguiente pantalla se puede apreciar cómo de una búsqueda obtenida de 342 muestras (ver captura de pantalla anterior), aparecen sólo las muestras que corresponden con el tipo tumoral definido NT (Solid Tissue Normal). Esto reduce de forma considerable las posibildades facilitando el prceso de selección de muestras.

Clase: corresponde al proyecto específico que se desea consultar. Actualmente, en el repositorio GDC no hay datos de todos los proyectos (al menos los de tipo Harmonized), y se espera que se vayan insertando en breve. Este es la razón de haber optado por una selección simple en lugar de una múltiple y haber incluido la opción de buscar todos los

proyectos TCGA; con la estrategia implementada si se consulta a proyectos TCGA y TARGET al mismo tiempo, y como hay proyectos TARGET sin datos de metilación la consulta se interrumpe y no sigue con el resto de proyectos indicados. Otra restricción aplicada es la búsqueda de datos clasificados como de tipo abierto; esto podría ser otra línea de trabajo futuro, la recueración de datos de tipo restringido para los que se necesita una cuenta de investigador y modificar ligeramente el filtro de acceso a los datos incluyendo soporte a la autenticación con GDC.

```
timeCheckFiles <- 60000 # 1 minuto

  observe({
    invalidateLater(timeCheckFiles, session)
    try({
      if (exists(fPlfms) && (exists(fprojGDC)))
        rm(fPlfms,fProjGDC)
      fPlfms <- as.data.frame(read.csv('data/platforms.csv',stringsAsFactors
= F,header=T))
      if (!exists('fPlfms'))
      fPlfms <- defGPL

      fprojGDC <- as.data.frame(read.csv('data/projGDC.csv',stringsAsFactors
= F,header=T))
      if (!exists('fprojGDC')){
        fprojGDC <- as.data.frame(TCGAbiolinks::getGDCprojects())[,c(6,4,5)]
        if(!exists('fProjGDC'))
          fprojGDC <- tGDC1
      }
    },silent=T)
  })
```

En el fragmento de código se puede observar el mecanismo implementado para cargar los datos de los proyectos almacenados en GDC, en caso de generarse nuevos proyectos en un futuro. El orden de prioridad para uso en la aplicación es:

1. Consulta online mediante función getGDCprojects()
2. Usar datos de fichero local projGDC.csv (con los proyectos a fecha de elaboración de este trabajo).
3. Tabla estática almacenada como parámetro en server.R → tGDC1

En el peor escenario en el que no exista ni el fichero y se produzcan errores de conexión, los datos se cargan de la tabla estática.

También se ha diseñado una ventana de referencia para las equivalencias entre el proyecto, tipo de tumor que estudia y sitio primario afectado.

Legacy (checkbox): el repositorio GDC de forma general puede verse como dos repositorios de datos en función de las validaciones realizadas. El *legacy archive*, implementado en la aplicación como

legacy = TRUE está contrastado con ensamblados del genoma más antiguos (hg18,hg19), y por tanto, en caso de descargar datos de estos estudios o sus anotaciones hay que tener esto en cuenta. En la aplicación web GDC se buscan en un apartado diferente (legacy archive). El resto de estudios (legacy=FALSE) se consideran adaptados al ensamblado hg38, y serían de tipo Harmonized. Otro dato que permite discriminar si un fichero procede de legacy archive o no es la pequeña variación en la categoría asignada. En los datos tipo harmonized los archivos tienen la categoría 'DNA Methylation', mientras que en los legacy varía a 'DNA methylation'. Esta sutileza es importante, ya que si no se tiene en cuenta y no se convierte a mayúsculas o se usa case.sensitive=F, pueden no producirse los resultados esperados.

El resto de criterios de GDC no van a afectar a los resultados de búsqueda, sino a las muestras recuperadas en la ventana correspondiente, y tras seleccionar los resultados. Para facilitar al usuario la selección, se ha realizado la siguiente aproximación al problema: el campo Nombre, de la tabla de resultados de los estudios GDC sufre un preproceso en el que se muestran sólo los tipos tumorales existentes en el proyecto. La idea es ofrecer algo más de información al usuario facilitando las búsquedas de forma anticipada, con un coste computacional ligeramente superior ya que antes de ofrecer los resultados hay que calcular los tipos de muestra de cada estudio.

Esta misma idea se lleva a cabo también con el campo plataforma, en el que se muestra al usuario las plataformas distintas de las muestras que conforman el estudio, de forma que puede haber estudios realizados con Illumina Human Methylation 27K, 450K o ambas, como se aprecia en la anterior captura de pantalla.

Tipo tumoral: selección de un tipo tumoral específo, y de acuerdo a una nomenclatura específica. Por ello, se ha incluido el una ventana de ayuda en el link '*Tipos...*' que muestra la equivalencia entre la abreviatura y el tipo de muestra (éste último es el que aparece en el campo nombre).

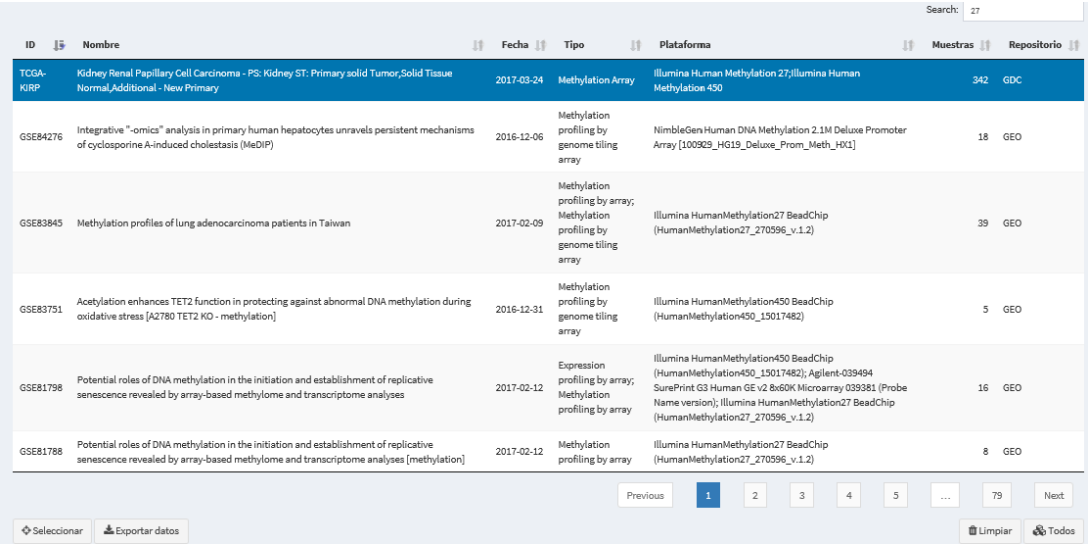
Sexo: basado en los posibles valores que pueden almacenarse en GDC Male, Female, Not Reported, Unknown, y teniendo en cuenta que no todas las muestras disponen de información clínica.
Raza: muestra los posibles valores de razas, además de Other, Not reported.

Etnia: los posibles valores son Hispanic Or Latino, Not Hispanic Or Latino, Not Reported
Tras haber descrito los tipos de elementos de entrada para fijar los criterios de búsqueda, se explica la naturaleza reactiva a las búsquedas de cada uno. Esto indica que algunos elementos producen cambios automáticos que activan los filtros de búsqueda, mientras que otros quedan a la espera de usar los botones 'manuales' de búsqueda

(botones buscar de sidebar con icono de lupa o botón debajo del elemento repositorio).

Elemento UI	*Reactividad*	*Motivo*
Keyword		Podría saturar las búsquedas con cada palabra insertada
Repositorio		Al seleccionar GDC o todos no daría tiempo a seleccionar el resto de filtros ocultos a priori
Organismo	X	Cambia también si se modifica la configuración
Plataforma	X	Cambia también si se modifica la configuración
Tipo	X	Cambia también si se modifica la configuración
Fecha	X	En cualquiera de los dos límites de fecha
Muestras	X	En cualquiera de los dos límites de número de muestras implicadas
Clase		Aplicar otros filtros de GDC
Legacy		Aplicar otros filtros de GDC
Tipo tumoral		No aplica a los filtros (pantalla de muestras)
Sexo		No aplica a los filtros (pantalla de muestras)
Raza		No aplica a los filtros (pantalla de muestras)
Etnia		No aplica a los filtros (pantalla de muestras)

El último aspecto a tener en cuenta para realizar las búsquedas es relativo al repositorio ArrayExpress, que tiene una redundancia con GEO. Esto afecta al diseño de la aplicación, cuyo filtro descarta en su última etapa los estudios que no se han registrado de forma específica; por tanto, si se desean buscar estudios sólo de ArrayExpress, la aplicación descarta este tipo de estudios redundantes.



Search: 27

ID	Nombre	Fecha	Tipo	Plataforma	Muestras	Repositorio
TCGA-KIRP	Kidney Renal Papillary Cell Carcinoma - PS: Kidney ST: Primary solid Tumor,Solid Tissue Normal,Additional - New Primary	2017-03-24	Methylation Array	Illumina Human Methylation 27;Illumina Human Methylation 450	342	GDC
GSE84276	Integrative "-omics" analysis in primary human hepatocytes unravels persistent mechanisms of cyclosporine A-induced cholestasis (MeDIP)	2016-12-06	Methylation profiling by genome tiling array	NimbleGen Human DNA Methylation 2.1M Deluxe Promoter Array [100929_HG19_Deluxe_Prom_Meth_HX1]	18	GEO
GSE83845	Methylation profiles of lung adenocarcinoma patients in Taiwan	2017-02-09	Methylation profiling by array; Methylation profiling by genome tiling array	Illumina HumanMethylation27 BeadChip (HumanMethylation27_270596_v.1.2)	39	GEO
GSE83751	Acetylation enhances TET2 function in protecting against abnormal DNA methylation during oxidative stress [A2780 TET2 KO - methylation]	2016-12-31	Methylation profiling by genome tiling array	Illumina HumanMethylation450 BeadChip (HumanMethylation450_15017482)	5	GEO
GSE81798	Potential roles of DNA methylation in the initiation and establishment of replicative senescence revealed by array-based methylome and transcriptome analyses	2017-02-12	Expression profiling by array; Methylation profiling by array	Illumina HumanMethylation450 BeadChip (HumanMethylation450_15017482); Agilent-039494 SurePrint G3 Human GE v2 8x60K Microarray 039381 (Probe Name version); Illumina HumanMethylation27 BeadChip (HumanMethylation27_270596_v.1.2)	16	GEO
GSE81788	Potential roles of DNA methylation in the initiation and establishment of replicative senescence revealed by array-based methylome and transcriptome analyses [methylation]	2017-02-12	Methylation profiling by array	Illumina HumanMethylation27 BeadChip (HumanMethylation27_270596_v.1.2)	8	GEO

Previous 1 2 3 4 5 ... 79 Next

Seleccionar Exportar datos Limpiar Todos

Ilustración 5: Tabla de resultados (estudios)

La tabla de resultados permite una selección de estudios con unos límites configurables en las opciones de hasta 5 estudios (por defecto se toman 3). La estructura de campos de la tabla es la misma para todos los repositorios y el diseño está basado en un sistema de filtros de forma que cada repositorio tiene un filtro asociado, basado en una estrategia de acceso específica e independiente a los demás, y que devuelve los resultados con esta estructura definida:

ID: es el identificador de cada repositorio. GSExxxx en el caso de GEO, E-MTAB-xxxx / E-MEXP /... en el caso de ArrayExpresss, TCGA-xxxx / TARGET-xxxx en el caso de GDC (al menos por ahora ya que hay previsión de que se incluyan nuevos proyectos).

Nombre: nombre del estudio. En el caso de GDC y como se ha mencionado se realiza un preproceso para mostrar los tipos tumorales implicados.

Fecha: la fecha de publicación en el repositorio. En el caso de GDC se toma la fecha de modificación de cualquiera de las muestras del proyecto. Esto hay que tenerlo en cuenta ya que casi todas las fechas serán bastante recientes, ya que sufre constantes actualizaciones.

Tipo: se indica la categoría que se le ha asignado en el repositorio correspondiente, teniendo en cuenta que independientemente del tipo asignado la aplicación realiza un proceso de marcado (flag oculto) para permitir seleccionar un tipo concreto en el filtro correspondiente.

Plataforma: hay que tener presentes dos cuestiones importantes. La primera, es que se realiza un preproceso para visualizar todas las plataformas implicadas en el estudio correspondiente y esta es una de las razones de que algunas consultas lleven más tiempo que otras (ejemplo, consultas GEO con organismo ‘Homo sapiens’). La segunda cuestión es relativa al repositorio ArrayExpress, que no muestra las

plataformas implicadas cuando se trata de estudios con perfiles de metilación obtenidos con plataformas de secuenciado. La razón parece estar en que dichos perfiles se almacenan como secuencias en otro repositorio externo (European Nucleotide Archive), y ArrayExpress se limita a indexar estos perfiles.

Flag: es el último valor asignado a los resultados, y que permanece oculto sin mostrarse en la tabla de resultados. También se usa para la obtención de estadísticas en la pantalla correspondiente.

Otras opciones de la tabla de resultados son la exportación de datos a fichero y permitir seleccionar todos los resultados mostrados o limpiar la selección realizada en la tabla. La selección de resultados es bastante útli cuando se han filtrado estudios con el campo 'Search:', habilitado en la parte superior derecha. En la captura anterior, se puede apreciar como los resultados se pueden acotar más con este campo, y lo que se muestra es mediante la cadena '27' es que busque entre los estudios recuperados los que contengan este valor en cualquiera de los campos (la idea de esta consulta es recuperar estudios con plataforma IlluminaHumanMethylation27k sin tener que cambiar al modo búsuqueda por plataforma).

Ahora se puede comprender el motivo de no haber incluido una búsqueda por identificador que podría resultar redundante. En los únicos casos que podría resultar útil es si no se conoce ningún dato del estudio excepto el código, bastante poco común. Por otro lado, si se quiere buscar un estudio de una plataforma en concreto, tras recibir los resultados se filtra la tabla con el campo 'search' y se inserta en este caso el identificador a localizar.

En definitiva, se puede observar cómo se pueden aplicar una serie de combinaciones entre los filtros de consulta a los repositorios combinado con un filtrado de la tabla obtenida de resultados . Una vez revisados los elementos de la interfaz y la estructura de datos de la tabla obtenida, se detallan las estrategias de acceso a los repositorios de los diferentes filtros. Para este fin, se muestran fragmentos de código que clarifican el diseño implementado, de las llamadas a dichos filtros

```
filtroGEO(keyWds,operador,iPlat,iSmps,iFech,iTipo,iOrga)
...
filtroAE(keyWds,operador,iSmps,iFech,iTipo,iOrga,iPlat)
...
filtroGDC(iSmps,iFech,iTipo,iClas,iLega)
```

Filtro GEO: recupera los criterios de posibles palabras clave, tipo de combinación de palabras clave (AND/OR), plataforma u organismo (depende de cómo se haya configurado), fecha de publicación o registro (también configurable desde las opciones) y tipo de estudio.

La estrategia de consulta es el acceso a una base de datos local SQLite con el diagrama entidad relación que se muestra en la siguiente captura:

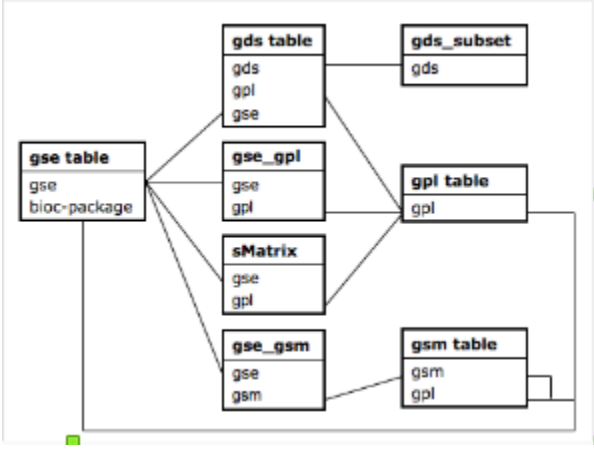


Ilustración 6: Diagrama entidad relación de GEOmetadb.sqlite

Las diferentes consultas que se realizan desde varias partes de la aplicación se basan en la función creada ad hoc que se muestra a continuación, y que también ofrece ventajas desde el punto de vista de la seguridad, ya que evita que queden conexiones abiertas a la base de datos, de forma que cada vez que se quiere hacer una consulta SQL se abre una conexión y se cierra tras obtener los datos. Esto podría tener inconvenientes en otro tipo de usos de la base de datos, pero aquí no se pueden producir incongruencias, ya que este sistema sólo realiza consultas (no modifica tablas, y al actualizar se actualiza la base de datos completa).

```
# Query GEO

fGEO <- function(sql){

  if(!file.exists('data/GEOmetadb.sqlite'))
    getSQLiteFile(destdir='data')
  con <- dbConnect(SQLite(),'data/GEOmetadb.sqlite')
  info <- dbGetQuery(con,sql)
  dbDisconnect(con)
  return(info)
```

En algunos casos se han usado consultas algo complejas con operadores JOIN, y son las que suponen el cuello de botella en todo el proceso.

El filtro realiza la obtención de datos, marcado y lo adapta a la tabla de resultados en función de los parámetros de la interfaz seleccionados, como el resto de los filtros.

Filtro ArrayExpress: usa exactamente los mismos parámetros (argumentos que el filtro anterior), con algunas modificaciones o procesos previos. El principal cambio es la conversión de la

nomenclatura usada para la plataforma (código GEO, GPLxxxx) a A-GEOD-xxxx, donde xxxx es equivalente en los dos repositorios.

La estrategia de adquisición en este caso es una query XML con parsing posterior de resultados (función extractXML() creada para tal efecto). Aquí se plantearon varias opciones, entre las que se planteó un acceso con JSON que presenta ventajas en la velocidad de procesamiento pero con una estructura de datos diferente.

El proceso se puede describir de la siguiente forma:

1. Construcción de la query con los parámetros (argumentos)
2. Parsing de la respuesta XML
3. Marcado y adaptación al formato de tabla para se coherente con el resto de resultados.

El cuello de botella está en la tarea de parsing, que es la que consume mayor tiempo computacional.
Filtro GDC:a diferencia de los anteriores no usa ni las palabras clave, ya que cada proyecto se considera como un todo, ni la plataforma (para estudios de metilación se usan pocas plataformas) ni el organismo, ya que todas las muestras proceden de humanos.

La estrategia que se sigue en este caso es el uso del API con la función *GDCquery()* que se encarga de devolver en forma tabular una lista de ficheros, que satisfacen los criterios del proyecto concreto y datos legacy o Harmonized.

Aquí habría que hacer una pequeña aclaración en cuanto la estructura de los datos de este repositorio, en el que está basado en dos tipos de identificadores: el de caso y el de fichero. La correspondencia entre ellos es de uno a muchos, de forma que puede haber un mismo caso del que se almacenen distintos ficheros; un ejemplo, sería una muestra de tumor de cáncer de próstata de un paciente concreto con cuatro ficheros de datos correspondientes a su secuenciación, hibridación en array 27K y/o 450K de metilación e hibridación en array de expresión, y cada uno de los ficheros requiere un pipeline de análisis diferente.

Una vez obtenidos los datos que se marcan y adaptan a la tabla de resultados, como el resultado de la consulta es un conjunto de ficheros que suelen corresponder con un fichero por muestra en el caso de la categoría de datos de metilación, esta tabla se mantiene en memoria para un filtrado posterior de las muestras según los fenotipos definidos en los filtros correspondientes.

La siguiente pantalla del módulo de búsqueda tiene la finalidad de realizar una selección de muestras eficiente.

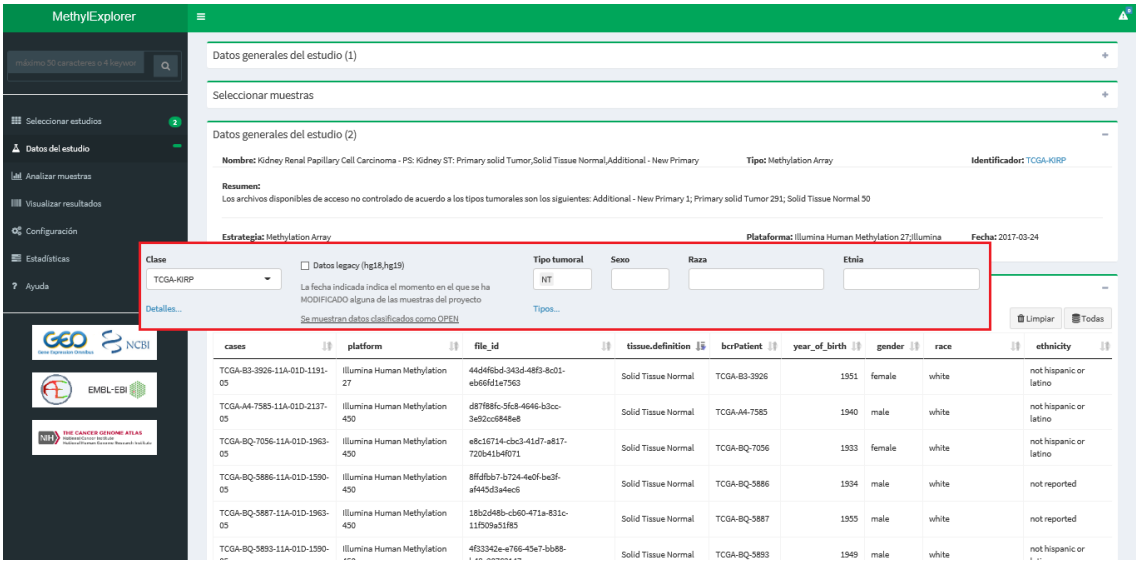


Ilustración 7: Interfaz de la pantalla de selección de muestras

El diseño de la interfaz permite mostrar hasta cinco estudios con sus tablas de selección de muestras correspondientes. Este límite de estudios mostrados que puede configurarse en el rango [1,5] se ha impuesto por dos motivos: evitar sobrecarga de datos en memoria, y no saturar demasiado esta pantalla de selección.

Detrás del diseño subyace un sistema de paneles ocultos que se muestran dependiendo de los estudios que se hayan seleccionado en la pantalla anterior. Para clarificarlo, se muestra un fragmento de código de los paneles (que engloba a los elementos box())

```
conditionalPanel(condition='input.d1Stud_rows_selected.length>0'
...
conditionalPanel(condition='input.d1Stud_rows_selected.length>1'
...
conditionalPanel(condition='input.d1Stud_rows_selected.length>2'
...
conditionalPanel(condition='input.d1Stud_rows_selected.length>3'
...
conditionalPanel(condition='input.d1Stud_rows_selected.length>4'
```

Otro elemento a destacar es el uso de cajas que pueden minimizarse o colapsar en la aplicación dependiendo de las necesidades del usuario. Este potencial se ha aprovechado para mostrar colapsadas las tablas de muestras, porque lo que se pretende es que el usuario compruebe que la descripción recuperada de los distintos estudios, así como estrategias de diseño u otros parámetros es la deseada antes de proceder a la descarga de datos de las muestras.

Se puede observar en la captura de pantalla, cómo sólo aparecen las muestras que corresponden al parámetro tipo tumoral NT, que corresponde al tipo ‘Solid Tissue Normal’.

Siguiendo con un recorrido por esta pantalla de selección se observan hasta diez elementos tipo box, en los que cada estudio está representado por dos de estos elementos, uno para completar la información del estudio y el otro para mostrar las diferentes muestras que componen el estudio.

Como se explicó en el apartado anterior, hay estudios que tienen muestras que pertenecen a varios repositorios, de ahí que se haya incluido el campo plataforma en esta tabla. Respecto a la información que se muestra del estudio se ha realizado la selección de los siguientes campos:

Nombre: el nombre que corresponde al de la tabla de resultados anterior.

Tipo: Indica el tipo de estudio en las categorías correspondientes. Si hay diferentes plataformas, es normal que aparezcan varios tipos.

Identificador: ID del repositorio específico. El identificador se muestra como un enlace al repositorio correspondiente. Para ello se ha creado una función específica, ya que cada repositorio tiene una estructura diferente de consultas.

```
urlConGEO <- 'https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc='
urlConAE <- 'http://www.ebi.ac.uk/arrayexpress/experiments/'
urlConGDC <- 'https://portal.gdc.cancer.gov/projects/'

 creaURL <- function(gse){

   url <- ''
   try({

     if (!is.null(gse)&&dim(gse)[1]>0){

       if (gse[,'Repositorio']=='GEO')
         url <- paste(urlConGEO,gse[,'ID'],sep='')
       if (gse[,'Repositorio']=='ArrayExpress')
         url <- paste(urlConAE,gse[,'ID'],sep='')
       if (gse[,'Repositorio']=='GDC')
         url <- paste(urlConGDC,gse[,'ID'],sep='')
     }
   },silent=T)

   return(as.character(url))
 }
```

Resumen: es la descripción del estudio. En el caso de GDC se ha decidido mostrar información relativa al número de muestras de cada tipo tumoral existente.

Estrategia: además del tipo de plataforma, en algunos casos suele dar información del diseño del estudio de las distintas muestras

Plataforma: muestra todas las plataformas relacionadas con las muestras del estudio (cada muestra obtenida con una plataforma).

Fecha: es la fecha de publicación en el repositorio. GEO y ArrayExpress también almacenan fechas de registro y actualización, pero esto no sucede en algunas muestras de GDC, de ahí que se muestre sólo esta fecha.

A continuación, se enumeran los campos que forman parte de la tabla de muestras, en las que se muestra una mayor flexibilidad, ya que no hay una correspondencia directa entre la forma de almacenar los fenotipos entre los tres repositorios. El elemento usado para el almacenamiento en memoria es una lista en lugar de una tabla o dataframe, lo que permite esta flexibilidad, que permite que cada repositorio tenga un conjunto variable de campos. En función del repositorio del que procede la muestra, se recuperan los siguientes campos:

GEO: se recupera el identificador de muestra (GSMxxxx), la plataforma usada, el nombre asignado, descripción y los dos posibles canales en caso de haberlos usado. En este punto hay que decir, que hay muestras registradas que usan los dos canales y otros usan sólo uno; esto depende del diseño de sondas de cada plataforma específica, y es el usuario el responsable de la selección de muestras, de forma que la aplicación no obliga a seleccionar muestras pareadas registradas como individuales (con código GSM único) pero que cada una corresponda a cada uno de los canales del mismo ensayo.

ArrayExpress: aquí la estrategia de consulta de muestras es diferente, ya que no se cuenta con una base de datos de consulta como en el caso anterior. Aprovechando el hecho de que los datos de este repositorio son compatibles con el formato MAGE-TAB, se descargan los archivos IDF y SDRF del estudio en cuestión, éste último el que se usa para extraer la información de las muestras. En concreto, se extraen los campos 'Source.name' y todos los campos que contienen la palabra 'Factor' y 'Label', que corresponden a las características de los fenotipos, y las marcas correspondientes a los colorantes empleados para la lectura de la hibridación (Green/Red). El problema es que diferentes experimentos tienen campos diferentes ('Factor_tissue','factor_age',...) y sería inviable realizar un tratamiento individualizado de los posibles factores.

GDC: con este repositorio se usa el identificador de caso, plataforma, identificador único de fichero (uuid), tipo tumoral, código del paciente al que pertenecen las muestras, año de nacimiento, género, raza y etnia. Como se observa en las capturas, en el código de muestra o caso viene implícito el prefijo que corresponde al código de paciente, al menos en los que proceden de proyectos TCGA.

Por último, y para finalizar con este módulo y pantalla, se han añadido botones de seleccionar todos y limpiar selección de las muestras en cada tabla. Al igual que se ha realizado con los estudios se ha fijado un

parámetro de sesión para evitar sobrecarga de datos que limita el número de muestras a analizar. Por tanto, la aplicación comprueba antes de pasar a la carga y análisis de muestras que se está dentro de este rango definido. Otra de las comprobaciones importantes es la comprobación de que todas las muestras seleccionadas, y con independencia al repositorio al que pertenezcan tengan nomenclaturas equivalentes. Se podrán combinar muestras procedentes de Illumina Human Methylation 27 y GPL8490, por ejemplo, ya que estas funciones comprueban que haya una equivalencia. Para clarificar este sistema se muestra una captura de pantalla.

	A	B	C	D	E	F
1	ID	Nombre	ID_AE	ID_GDC	Tipo	ANNOTATION
2	GPL8490	Illumina HumanMethylation27 BeadChip (HumanMethylation27_270596_v.1.2)	A-GEOD-8490	Illumina Human Methylation 27	A	IlluminaHumanMethylation27k.db
3	GPL13534	Illumina HumanMethylation450 BeadChip (HumanMethylation450_15017482)	A-GEOD-13534,A-MEXP-2255	Illumina Human Methylation 450	A	IlluminaHumanMethylation450k.db
4	GPL16304	Illumina HumanMethylation450 BeadChip [UBC enhanced annotation v1.0]	A-GEOD-16304,A-MEXP-2255	Illumina Human Methylation 450	A	IlluminaHumanMethylation450k.db
5	GPL18809	Illumina HumanMethylation450 BeadChip (v1.2, extended annotation)	A-GEOD-18809,A-MEXP-2255	Illumina Human Methylation 450	A	IlluminaHumanMethylation450k.db

Ilustración 8: Estructura del fichero platforms.csv

La función principal del fichero es establecer una base para la aplicación de los datos que se permiten analizar, y la función secundaria es almacenar la anotación de dicha plataforma (paquete Bioconductor), que se usa en la parte del análisis y en futuras ampliaciones de la aplicación. Por tanto, en este punto es donde se aprecia una limitación de funcionalidad, ya que si no se seleccionan plataformas equivalentes, no se puede continuar con los análisis. Hasta esta pantalla se pueden consultar muestras de estudios procedentes de secuenciación o de otros microarrays no habilitados.

También en este punto comienza a hacerse uso del mecanismo de notificaciones implementado en el header en la parte derecha. De forma que pueden aparecer los siguientes mensajes:

- *Plataformas no permitidas (warning)*: indica que al menos una de las plataformas seleccionadas no está en ninguna de las filas ni columnas del fichero, es decir, no se permite.

- *Plataformas no equivalentes (warning)*: indica que hay al menos una discordancia entre el campo plataforma de las muestras. Es decir, no están en la misma fila del fichero.

Para pasar a la descarga y análisis de las muestras, la aplicación comprueba primero si todas las plataformas de las seleccionadas están en el fichero (permitidas) con la función *checkPerm()* y después si son equivalentes con la función *checkEquiv()*.

El siguiente paso tras obtener la selección de muestras es la descarga y comprobación de los datos que almacenan. Este punto es muy importante, ya que se realiza en una misma etapa. Esto tiene la ventaja de mostrar un alto grado de transparencia, pero el inconveniente de que si no se conoce a priori el tipo de datos almacenados o la aplicación no es capaz de recuperarlos de maner automática, es el usuario el encargado de cargarlos mediante fichero externo. Esta es una de las

razones de crear enlaces a los estudios en la pantalla de selección de muestras en los que el usuario puede consultar si los datos almacenados son sólo de tipo RAW y así evitar comprobaciones de la aplicación, porque como se ha indicado, la aplicación cuenta con capacidad de gestionar datos que han sido procesados.

Módulo de **análisis**: es el encargado de realizar los análisis definidos de filtrado por niveles de metilación beta, búsqueda de regiones DMR o anotación de sondas. El aspecto que presenta este módulo en la ventana de 'Analizar Muestras' es el siguiente.

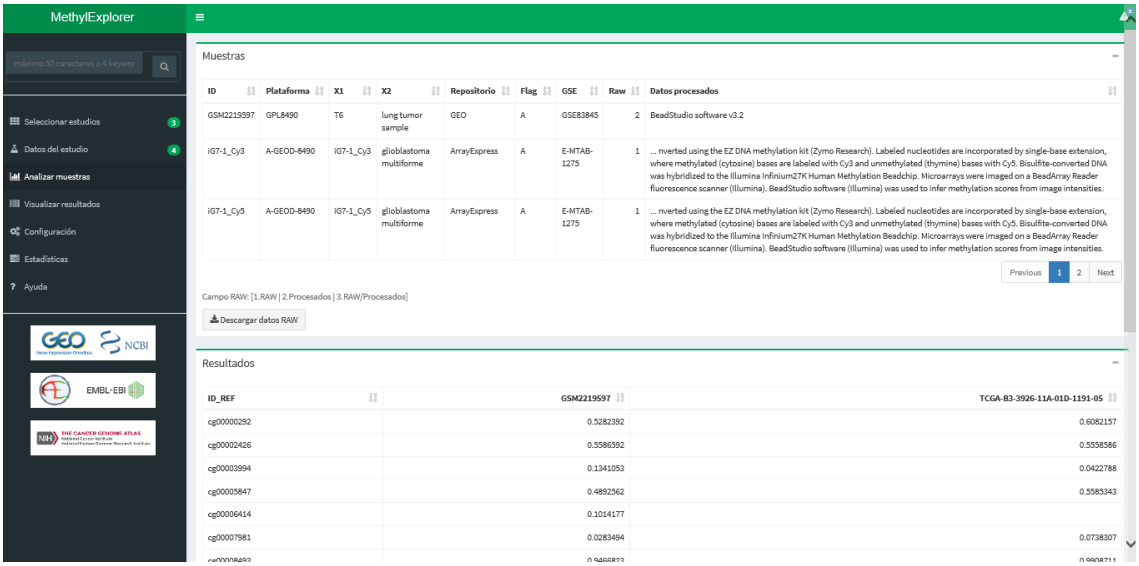


Ilustración 9: Interfaz de la pantalla de análisis

Se pueden apreciar cuatro bloques principales en este módulo, y con funciones diferentes:

Elemento **box de información de muestras** y posible descarga de datos RAW, para usar con otras aplicaciones. La función principal es recopilar información de los tipos de datos almacenados en las muestras, de forma que los procesados se cargan de forma directa. Dependiendo del repositorio en cuestión, las formas de obtener esta información también son diversas. La función principal implicada es *descargaMuestras()*

- GEO: se usa la función *getGEO(),* que carga información del fichero SOFT del experimento. Una vez descargado se consultan distintos elementos del header para determinar si hay datos RAW y el tipo de preprocesamiento al que se han sometido los datos.

- ArrayExpress: se consultan los ficheros IDF y SDRF (se vuelven a descargar ya que tienen un tamaño reducido), y se extrae la información tanto del protocolo completo del estudio (para mostrar los últimos caracteres donde suele aparecer el tipo de software con el que se han

preprocesado los datos a no ser que sean sólo RAW), como de si se disponen de datos RAW, y donde localizarlos. También se toma la ubicación del nombre del fichero de los datos procesados para cargarlos en la tabla

- GDC: la estrategia de descarga de datos varía respecto a las anteriores, ya que no se comprueban y descargan todas las muestras de forma indifivual sino que al tener los identificadores de fichero únicos, se usan éstos para descargarlos de forma simultánea, descomprimirlos y cargarlos en la tabla.

En los tres casos los datos recuperados se adaptan a una tabla de información con los siguientes campos:

ID: contiene el identificador de muestra: GSMxxx (GEO), nombre de la muestra (ArrayExpress) o código de caso (GDC).

Plataforma: la misma que en la pantalla anteriores

X1: Características de la muestra, en el caso de GDC se almacena el uuid del fichero

X2: Características de la muestra, en el caso de GDC se almacena el tipo tumoral.

Estos dos últimos campos tienen esta nomenclatura debido a la flexibilidad en los campos almacenados para las muestras (en lista), que pueden no tener relación entre ellos. X2 indica con mayor precisión el fenotipo de las muestras, siendo en el caso de GEO el descriptor de la muestra.

Repositorio: procedencia de la muestra/estudio.

Flag: en este caso se muestra el tipo de estudio, con las categorías establecidas por la aplicación (A,S,AS) ante una futura expansión en la funcionalidad.

GSE: Identificador del estudio de procedencia.

Estos tres últimos campos pueden servir como referencia en algunos casos, aunque siempre se mantienen en memoria los estudios seleccionados y muestras a los que se puede volver hacia atras volviendo a las pantallas correspondientes.

Raw: Es un sistema de codificación propio diseñado para indicar el tipo de datos disponible en cada muestra. En la captura de pantalla anterior se puede apreciar cómo sólo se cargan los datos procesados en la tabla porque son los únicos disponibles. Las dos muestras de las cuatro seleccionadas que contienen datos sólo

RAW se indican por si el usuario quisiera descargarlas para un uso con otra aplicación.

Datos procesados: extrae la información del tipo de preproceso sufrido por los datos, normalmente indica el tipo de software empleado. Es importante resaltar, que en el repositorio ArrayExpress hay que consultar el protocolo completo de la muestra en el IDF, y al ser un campo muy extenso se almacenan sólo los últimos 500 caracteres, ya que en última parte es donde se describe este preproceso indicado.

En este box de información, también se ha habilitado un botón para acceder que comprueba si las muestras cargadas disponen de datos RAW (campo Raw), y en los casos en que su valor es 1 o 3 procede a su descarga en una carpeta específica dentro (/ext/RAW).

El **segundo box** de este módulo y pantalla de análisis almacena los datos procesados cargados así como los posibles análisis a realizar.

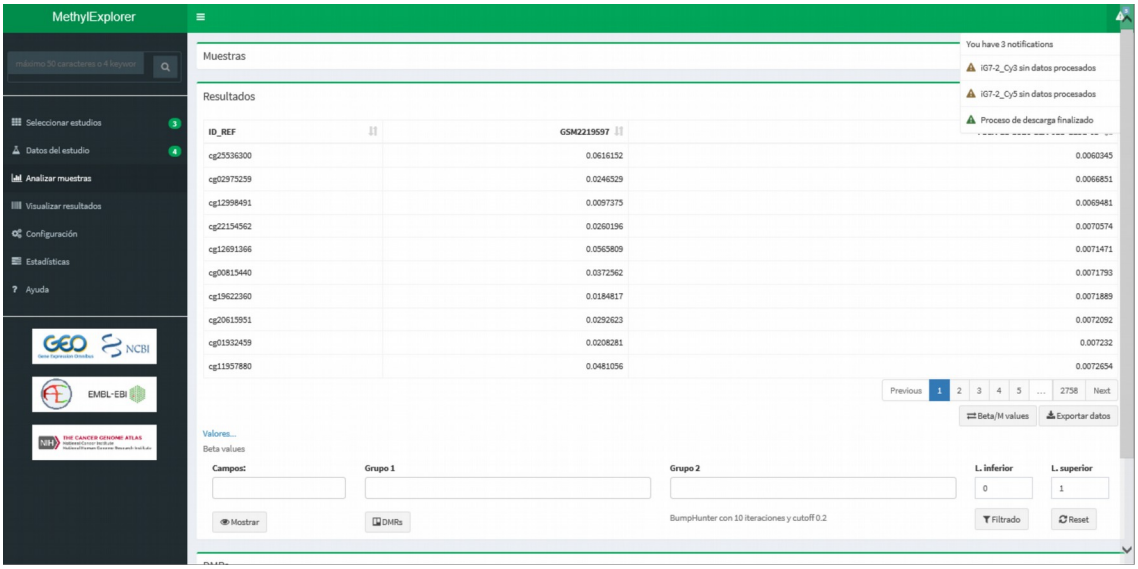


Ilustración 10: Elemento box del módulo de análisis

Una vez llegados a este punto en el que se cargan de forma exitosa las muestras que disponen de datos RAW, hay que explicar que los datos están almacenados en un objeto de tipo ***RatioSet*** del paquete minfi, y que almacena los valores beta y M de los datos. Además, en el proceso de identificación de DMRs, una vez definidos los grupos, estos datos se convierten al tipo ***GenomicRatioSet*** en el que además tienen las posiciones genómicas de cada sonda, definidas por cromosoma, comienzo y fin de secuencia.

Para realizar esta función es necesaria la carga de las anotaciones correspondientes que, en este caso se han tomado de ficheros de GDC al que se le han eliminado la columna de valores beta de la muestra. Es una de las posibles opciones, otra es la de descargar el dataset de

anotaciones que se especifica en el fichero (fila correspondiente), y mantenerlo en memoria según se necesite.

Este fichero no tiene las secuencias completas, pero para la funcionalidad básica de la aplicación es adecuado. Se muestra a continuación el fichero de la plataforma 27K, que se adjunta con los entregables, y que corresponde a un proceso de liftover desde el ensamblado hg19 al hg38.

B	C	D	E	F	G	H	I	J	K
Chromosome	Start	End	Gene_Symbol	Gene_Type	Transcript_ID	Position_to_T	CGI_Coordina	Feature_Type	
cg00000292	chr16	28878779	28878780	ATP2A1;ATP2	protein_coding	ENST000003	373;290;-1275	CGI:chr16:288	N_Shore
cg00002426	chr3	57757816	57757817	SLMAP;SLMA	protein_coding	ENST000002	1585;368;261;	CGI:chr3:5779	S_Shore
cg00003994	chr7	15686237	15686238	MEOX2	protein_coding	ENST000002	576	CGI:chr7:1639	.
cg00005847	chr2	176164345	176164346	AC009336.19;	protein_coding	ENST000004	13259;267;34	CGI:chr2:1761	N_Shore
cg00006414	chr7	149125745	149125746	RN7SL521P;Z	misc_RNA;pro	ENST000004	242;-672;602;	CGI:chr7:1491	N_Shore
cg00007981	chr11	94129428	94129429	PANX1;PANX	protein_coding	ENST000002	499;498	CGI:chr11:941	Island
cg00008493	chr14	93347431	93347432	COX8C;UNC7	protein_coding	ENST000003	239;14211	CGI:chr14:933	Island
cg00008713	chr18	11980954	11980955	IMPA2;IMPA2	protein_coding	ENST000002	-475;-563;-72;	CGI:chr18:119	Island
cg00009407	chr14	88824577	88824578	TTC8;TTC8;T	protein_coding	ENST000003	-80;423;-80;-8	CGI:chr14:888	Island
cg00010193	chr4	1151428	1151429	AC092535.3;T	antisense;unit	ENST000004	55;2299	CGI:chr4:1144	.
cg00011459	chr16	8796568	8796569	PMM2;PMM2	protein_coding	ENST000002	-1250;-1275;-1	CGI:chr16:879	N_Shore
cg00012199	chr14	20682865	20682866	ANG;RNASE4	protein_coding	ENST000003	-1313;-1236	CGI:chr14:206	Island
cg00012386	chr1	227734811	227734812	JMJD4;JMJD	protein_coding	ENST000003	601;601;-439;	CGI:chr1:2277	N_Shore
cg00012792	chr6	8064260	8064261	BLOC1S5;BLO	protein_coding	ENST000002	134;155;155;1	CGI:chr6:8063	Island
cg00013618	chr22	22026441	22026442	IGLVI-70;PR	IG_V_pseudo	ENST000005	364;17494;17	CGI:chr22:219	.
cg00014085	chr2	85354382	85354383	ELMOD3;ELM	protein_coding	ENST000003	-422;-376;-339	CGI:chr2:8535	N_Shore
cg00014837	chr12	6648091	6648092	ACRBP;ACRB	protein_coding	ENST000002	-630;-679;-679	CGI:chr12:664	S_Shore
cg00015770	chr4	121380852	121380853	QRFPR;QRFP	protein_coding	ENST000003	-172;208;-204;	CGI:chr4:1213	Island
cg00016968	chr1	112707826	112707827	RHOC;RHOC;	protein_coding	ENST000002	-698;-422;-746	CGI:chr1:1127	S_Shore
cg00019495	chr4	56681359	56681360	HOPX;HOPX;	protein_coding	ENST000003	541;349;348;-	CGI:chr4:5665	.
cg00020533	chr6	35513139	35513140	TULP1;TULP1	protein_coding	ENST000002	-200;-262;-117	CGI:chr6:3551	S_Shore
cg00021527	chr17	35809176	35809177	AC015849.12;	lincRNA;prote	ENST000006	-278;95384;-3	CGI:chr17:358	Island
cg00022606	chr20	461801	461802	TBC1D20;TBC	protein_coding	ENST000003	753;731;743	CGI:chr20:461	N_Shore
cg00022866	chr11	64340968	64340969	CCDC88B;CC	protein_coding	ENST000003	744;744;-639	CGI:chr11:643	N_Shore

Ilustración 11: Fichero de anotaciones usado en la aplicación para muestras con 27K

Observando la interfaz se intuyen las distintas opciones de análisis que se ofrecen, y que se detallan a continuación:

- *Opción de filtrado*, para permitir que puedan descartarse sondas que no entren en el rango establecido.

- *Mostrar campos*, para cargar otras columnas además de los identificadores de las sondas. Los posibles valores se cargan de los datos de anotaciones.

- *Conversión beta/M values*: permite conversión entre ambos valores manteniendo 7 cifras decimales. Cada uno tiene sus ventajas, los beta values tienen un rango mejor interpretable desde el punto de vista biológico, y los M values mayores cualidades estadísticas.

Las tablas que recogen los resultados del análisis DMR, y los datos procesados, que registran el filtrado, así como los procesos de conversión tienen opción de descarga externa, que podrían usarse con otras aplicaciones

El **tercer** elemento box recoge el resultado del análisis DMR entre las muestras, y almacena las posiciones genómicas y clusters identificados

en la salida del algoritmo Bumphunter, que se ha configurado con 10 iteraciones por motivos de rendimiento fundamentalmente.

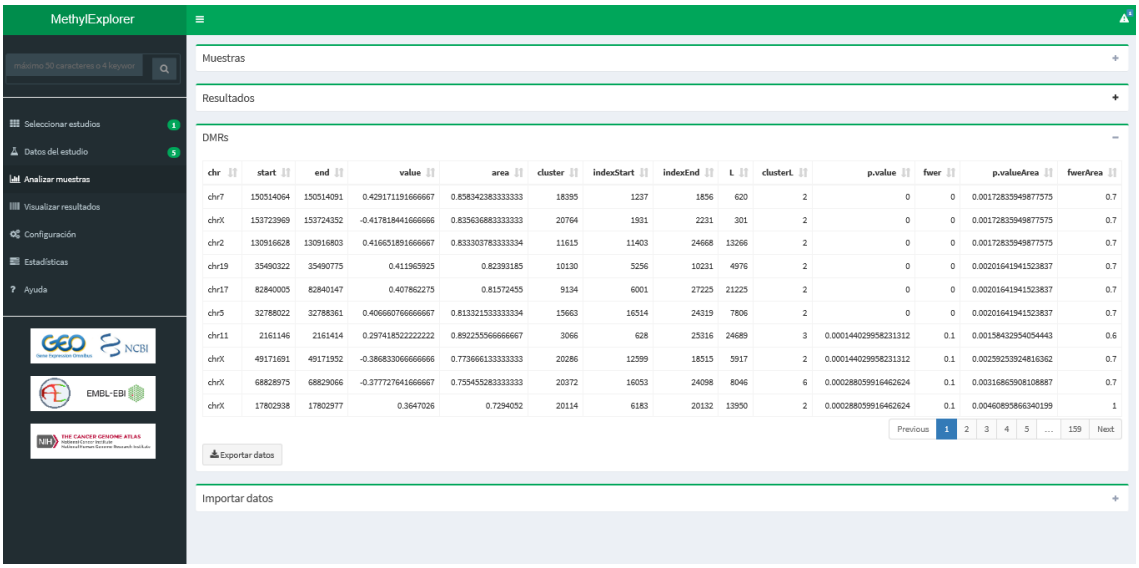


Ilustración 12: Elemento box de análisis con resultados de una comparativa de muestras

Los resultados se muestran ordenados por significancia (de mayor a menor), y como se ha indicado los datos pueden exportarse a un fichero para tratamiento posterior.

El **cuarto** y último box responde a la necesidad de una carga manual o de datos externos que maneje el usuario en la plataforma analizada. La interfaz de usuario es la que se muestra a continuación:

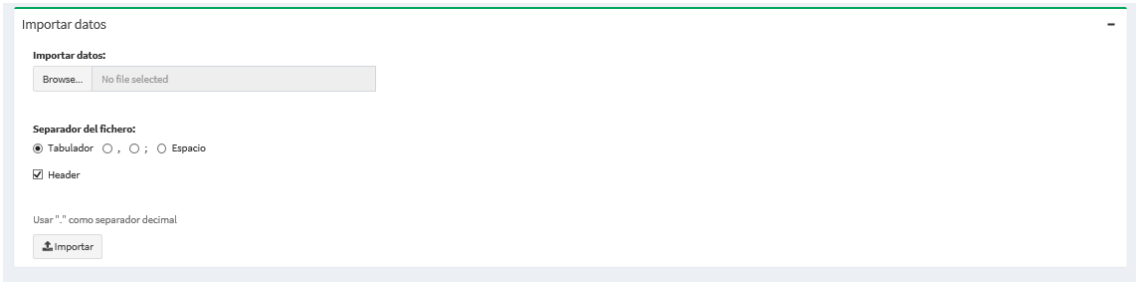


Ilustración 13: Elemento box de carga manual de ficheros

El usuario es reponsable de cargar unos datos adecuados, aunque es la aplicación la que realiza unas comprobaciones básicas: que el formato de los campos sea numérico, y que se cargue en la primera columna identificadores de sonda que comienzan por cg... para las plataformas Illumina admitidas.

Módulo de **visualización**: es el encargado de cargar las gráficas que permiten al usuario de una manera rápida el estado global de metilación de las muestras. Esta es la parte más susceptible a actualizaciones, mediante la inclusión de distintas gráficas: heatmap, gráficas tipo lollipop, o identificando los datos de metilación en su contexto genómico.

Ilustración 14: Interfaz de visualización de los datos procesados

La elección realizada de gráficas es la que se muestra en pantalla, y permite ver de forma global el grado de metilación. Se ha incluido un gráfico de densidad, una gráfica boxplot equivalente del total de muestras y un boxplot en función de la asignación de grupos realizada en la pantalla anterior. La base para la representación de estas gráficas son los datos cargados en el box de resultados que puede haber sido filtrado o no.

En concreto, si se realize una selección de sondas con valores beta en el rango [0,0.5], sobre los datos mostrados en la captura anterior, se pueden analizar diferencias en este rango de valores concreto y de forma global.

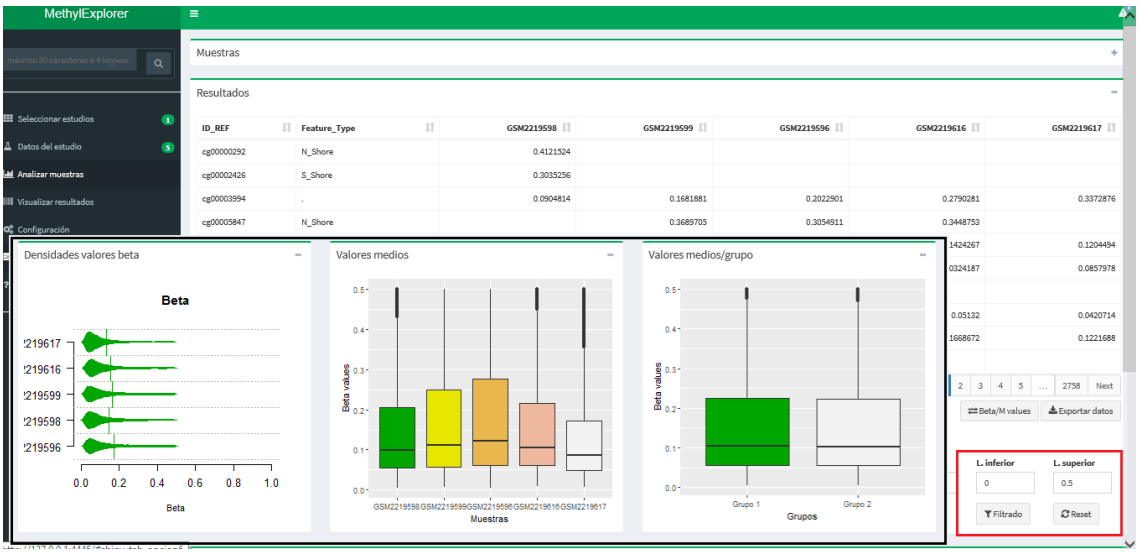


Ilustración 15: Visualización de un rango concreto de valores beta de forma global

Una vez realizado el recorrido por las pantallas de los módulos principales, se describen el resto de interfaces de las que consta la aplicación. La siguiente pantalla en el orden en el que se ha establecido en el menú es la de configuración, donde se pueden definir los parámetros de sesión de la aplicación.

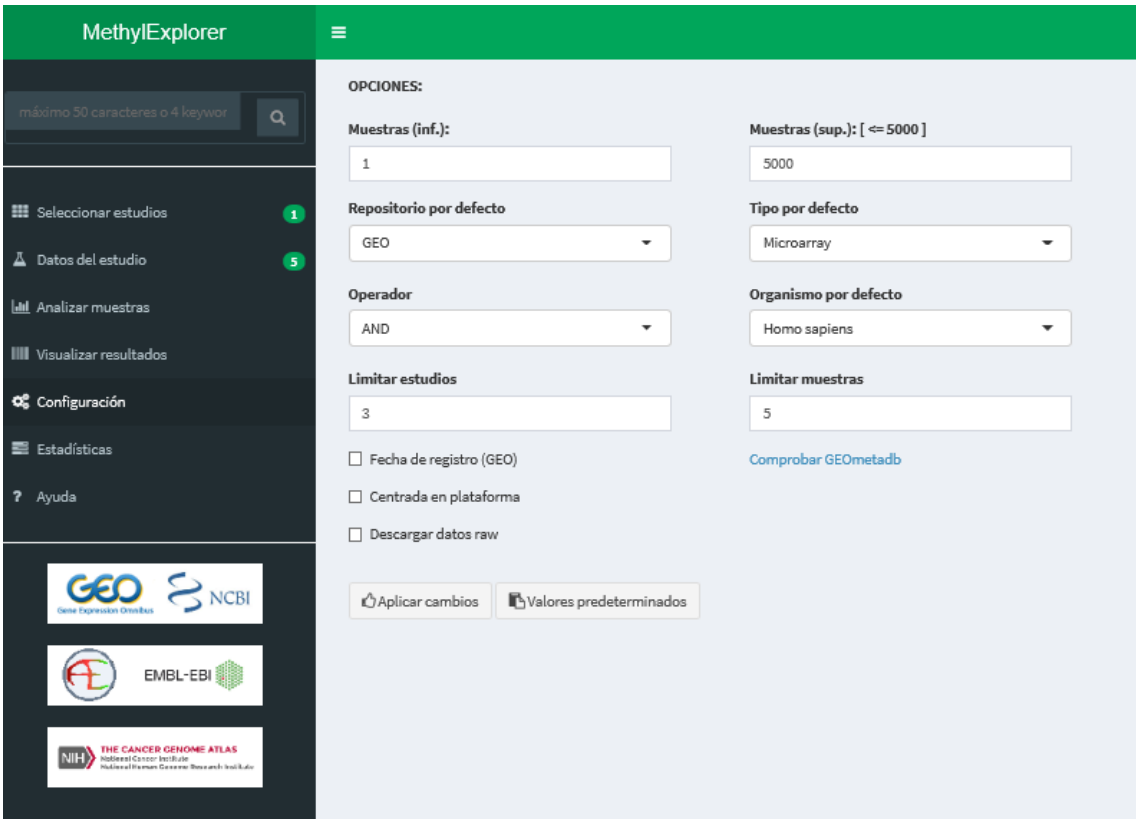


Ilustración 16: Interfaz de configuración

Con la excepción del enlace a la comprobación de la fecha de actualización de la base de datos para las consultas GEO, el resto de opciones corresponden a parámetros que personalizan los criterios de búsqueda, y que se almacenan de forma interna en forma de valores reactivos:

Límite de muestras inferior: no puede ser inferior a 1 ni superior al límite inferior.

Límite de muestras superior: se establece como 5000, ya que se han observado estudios de más de tres mil muestras, y no puede ser inferior al mínimo ni superior a este valor que se toma como valor máximo no superable.

Repositorio por defecto: el que el usuario quiere que aparezca en la pantalla de selección de estudios.

Tipo por defecto: análogo al anterior.

Plataforma por defecto: está oculta en las opciones, y es accesible al habilitar la opción centrada en plataforma.

Operador: tiene dos posibles valores OR o AND. Son los operadores que aplicará por defecto al introducir keywords de búsqueda, y como se intuye el operador AND es el más restrictivo. El único repositorio al que no afecta es GDC, cuyo filtro descarta este criterio.

Organismo por defecto: análogo a repositorio y tipo.

Limitar estudios: establece el máximo de estudios que puede gestionar la aplicación, y se establece sobre todo por cuestiones de rendimiento.

Limitar muestras: establece el máximo de muestras que se pueden gestionar. También establecido por cuestiones de rendimiento.

Fecha de registro (checkbox): permite buscar por fecha de registro (submission). Con algunos estudios pueden pasar meses desde la fecha de registro hasta la de publicación, y habría que tenerlo en cuenta.

Centrada en plataforma (checkbox): es la opción que permite habilitar el elemento de búsqueda por plataformas o por organismo.

Descargar datos RAW (está DESHABILITADA): se ha mantenido porque era una de las posibles mejoras, consistente en el tratamiento de datos raw y su preprocesamiento previo al análisis, funcionalidad que no se ha podido incorporar.

La siguiente tabla constituye a modo de resumen las opciones junto a sus variables y valores iniciales en la aplicación.

Opción	***Variable***	***Valor inicial***
Límite de muestras inf.	control$gMinSp	1
Límite de muestras sup.	control$gMaxSp	5000
Repositorio por defecto	control$gDefRp	GEO
Plataforma por defecto	control$gDefpf	Primera plataforma del fichero
Tipo por defecto	control$gDefTp	Microarray
Operador	control$gSchop	AND
Organismo por defecto	control$gOrgan	Homo sapiens
Limitar estudios	control$gCompa	3
Limitar muestras	control$gMaxMu	5
Fecha de registro	control$gDefFe	F
Centrada en plataforma	control$gOrPla	F
Descarga de datos RAW	control$gRawDa	DESHABILIDATA (F)

El resto de variales reactivas lo constituyen las siguientes:

isBeta: indica si los datos procesados mostrados son beta o M values.

GdefGD: almacena la tabla de proyectos GDC que se puede provenir de las siguientes fuentes ordenadas por prioridad de mayor a menor: fichero local (detecta cambios), consulta online mediante función, tabla estática cargada al inicio de la aplicación.

También quedaría añadir algunas variables de sesión:

defSmpSel: con el valor límite máximo del slider de la pantalla de selección de estudios.

DefGPL: que en caso de fallo de carga del fichero de plataformas, al menos, permite análisis de muestras relacionadas con la plataforma Illumina HumanMethylation 450K.

La siguiente pantalla corresponde a las estadísticas de los estudios recuperados, y se puede observar en la siguientes capturas de pantalla, ya que se ha establecido un menú con diferentes pestañas en función de los datos que se desean observar:

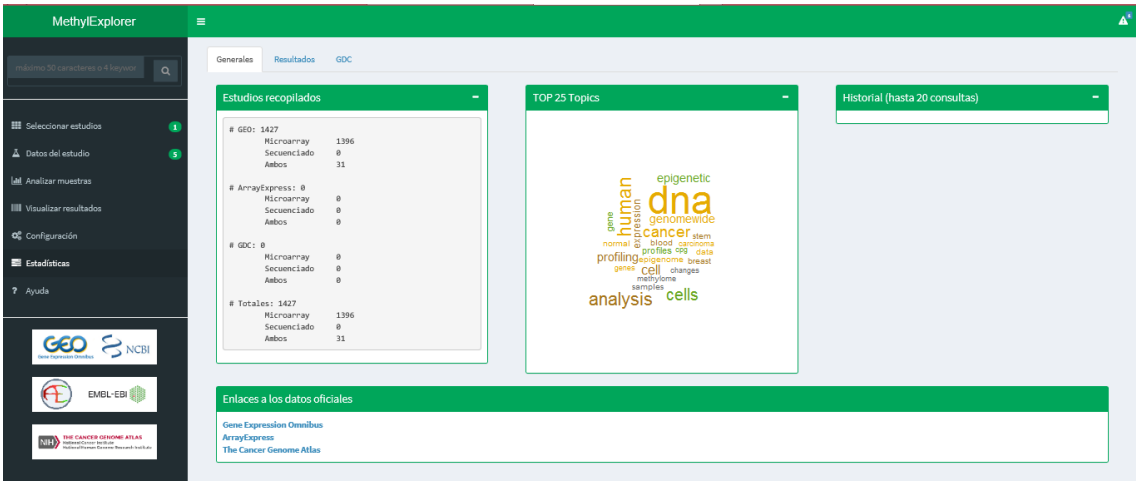


Ilustración 17: Interfaz de estadísticas: pestaña de datos generales

El primer elemento box muestra las distintos tipos de estudios recuperados en la consulta realizada. El sistema usado para identificarlos es el flag de marcado basado en las categorías de cada repositorio.

El gráfico del segundo box es del tipo wordcloud con las 25 keywords más frecuentes de los títulos de los estudios recuperados. Esto da una idea de globalidad.
En el tercer box está plegado el historial de búsquedas con las palabras clave almacenadas en memoria usadas en la sesión.

El último box inferior contiene unos enlaces a las estadísticas generales de los repositorios tratados, no sólo de estudios con datos de metilación.

La siguiente pestaña muestra estadísticas del número de muestras de los distintos resultados, según el repositorio de procedencia. La siguiente captura corresponde a resultados obetnidos de una consulta a los tres repositorios:

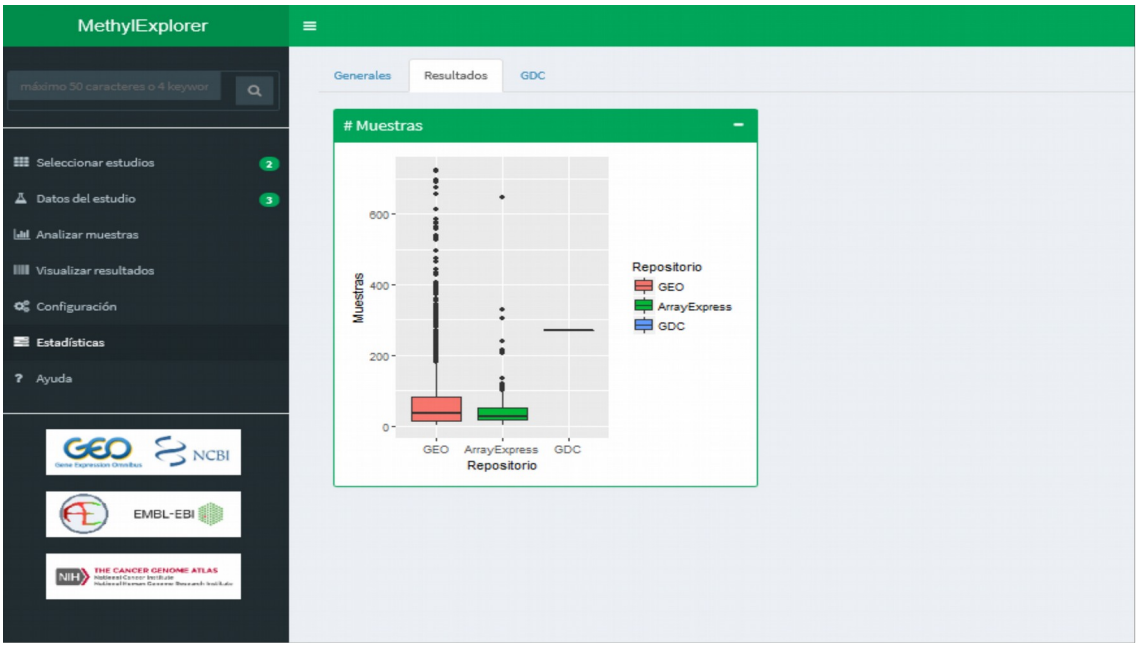


Ilustración 18: Interfaz de estadísticas: pestaña de resultados

La última pestaña muestra gráficas del repositorio GDC. En caso de no haber obtenido datos de este repositorio, sólo se mostrarían las gráficas superiores, y los elementos box inferiores aparecerían colapsados.

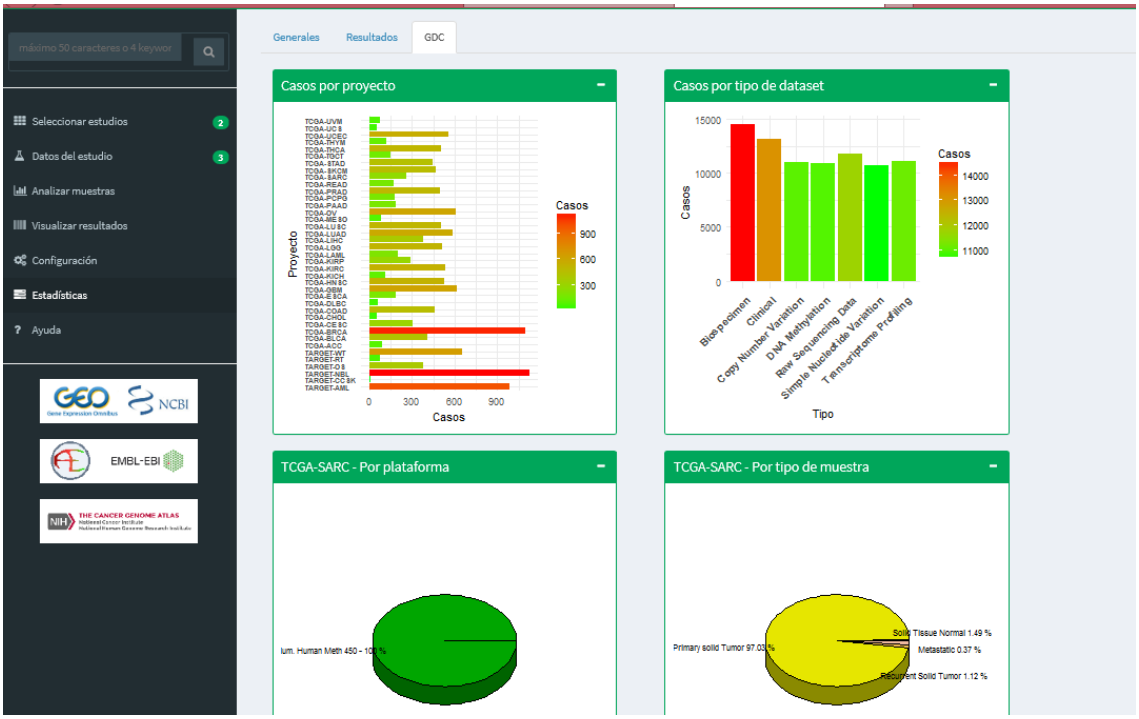


Ilustración 19: Interfaz de estadísticas: pestaña GDC

Para finalizar el recorrido por las diferentes interfaces de usuario, se muestra una captura de la pantalla de ayuda, donde se resumen las tipologías de elementos de información e interacción empleados y su posible reactividad en las consultas.

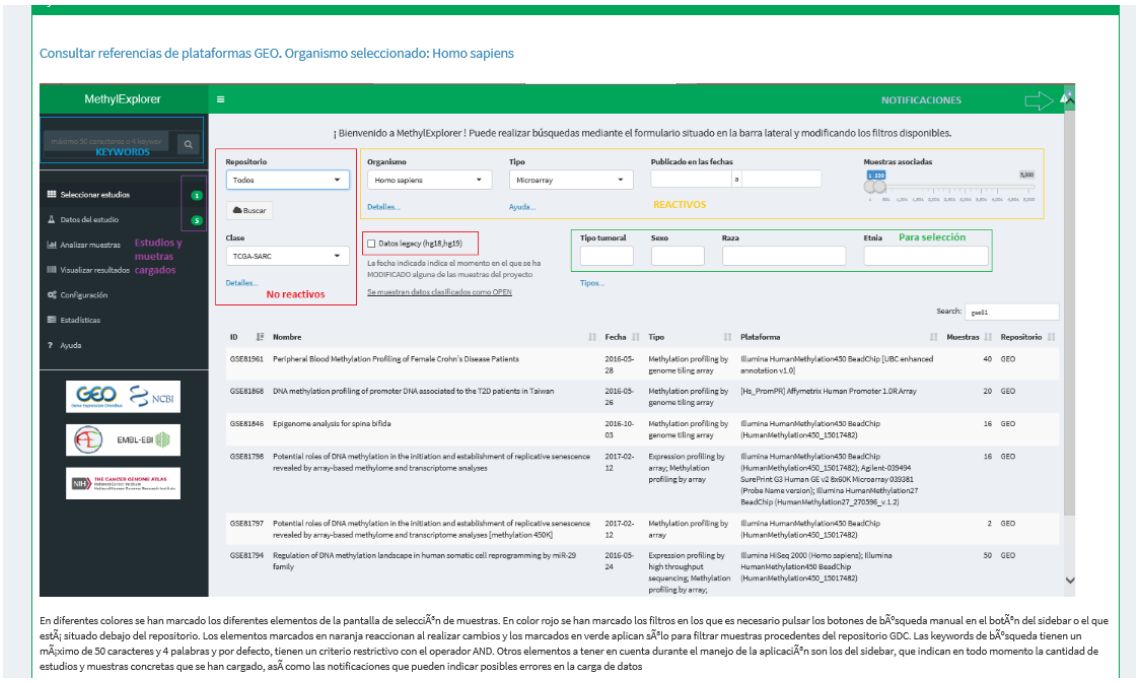


Ilustración 20: Interfaz de ayuda

Por último quedaría mostrar las tablas que se han diseñado para mostrar información al usuario en las diferentes pantallas. No se muestran las que se han mencionado anteriormente de estudios, muestras, datos procesados y análisis DMR, sino que son tablas de ayuda accesibles con links en lugar de botones para distinguirlas.

Detalles de plataformas: está integrada en la pantalla de selección de estudios, y depende de la plataforma que está cargada en ese momento.

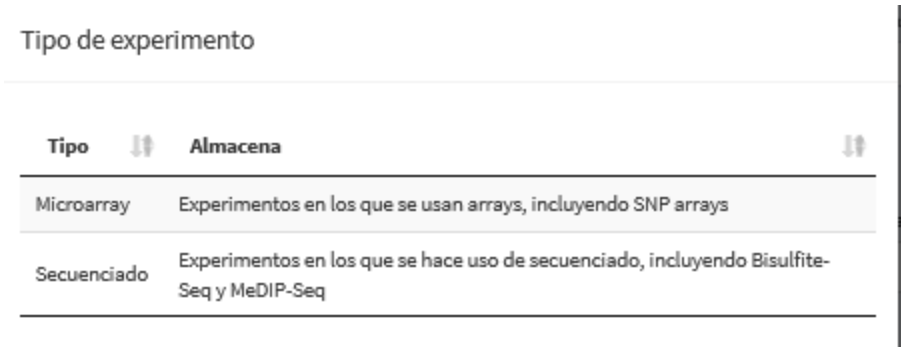

Detalles de las plataformas

ID	Nombre	Tipo	Organismo	Fabricante	Bioconductor
GPL8490	Illumina HumanMethylation27 BeadChip (HumanMethylation27_270596_v.1.2)	oligonucleotide beads	Homo sapiens	Illumina, Inc.	IlluminaHumanMethylation27k

Previous 1 Next

Ilustración 21: Detalles de plataformas

Tipos de experimento: explica las categorías generales que distingue la aplicación.

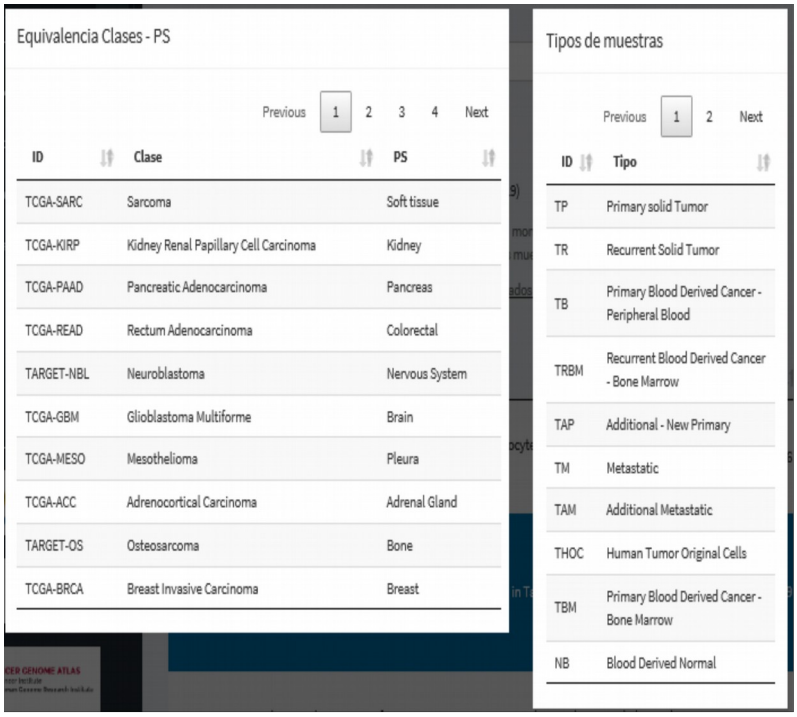

Tipo de experimento

Tipo	Almacena
Microarray	Experimentos en los que se usan arrays, incluyendo SNP arrays
Secuenciado	Experimentos en los que se hace uso de secuenciado, incluyendo Bisulfite-Seq y MeDIP-Seq

Ilustración 22: Tipos de experimento

Detalles de clase (proyecto): muestra la clase de los proyectos GDC, así como el tipo de cáncer en el que se centran y el lugar primario donde se desarrolla.

Tipos tumorales: indica la equivalencia entre la abreviatura usada para las muestras y el tipo tumoral al que alude.



Equivalencia Clases - PS

Previous 1 2 3 4 Next

ID	Clase	PS
TCGA-SARC	Sarcoma	Soft tissue
TCGA-KIRP	Kidney Renal Papillary Cell Carcinoma	Kidney
TCGA-PAAD	Pancreatic Adenocarcinoma	Pancreas
TCGA-READ	Rectum Adenocarcinoma	Colorectal
TARGET-NBL	Neuroblastoma	Nervous System
TCGA-GBM	Glioblastoma Multiforme	Brain
TCGA-MESO	Mesothelioma	Pleura
TCGA-ACC	Adrenocortical Carcinoma	Adrenal Gland
TARGET-OS	Osteosarcoma	Bone
TCGA-BRCA	Breast Invasive Carcinoma	Breast

Tipos de muestras

Previous 1 2 Next

ID	Tipo
TP	Primary solid Tumor
TR	Recurrent Solid Tumor
TB	Primary Blood Derived Cancer - Peripheral Blood
TRBM	Recurrent Blood Derived Cancer - Bone Marrow
TAP	Additional - New Primary
TM	Metastatic
TAM	Additional Metastatic
THOC	Human Tumor Original Cells
TBM	Primary Blood Derived Cancer - Bone Marrow
NB	Blood Derived Normal

Ilustración 23: Detalles de proyectos y tipos tumorales

Valores de las muestras: indican el tipo de dato procesado, e indica si los datos son beta values o m values.

Valores de intensidad de las muestras

ID	Valor
GSM2219597	'''Average Beta'''
GSM2219598	'''Average Beta'''
GSM2175410	'''normalized beta values'''

Previous 1 Next

Ilustración 24: Tipo de datos procesados

Comprobación de la fecha de la base de datos SQLite de GEO

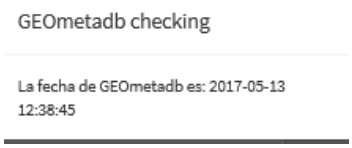


Ilustración 25: Fecha BD

Consulta de plataformas: carga las plataformas asociadas al organismo cargado en ese momento.

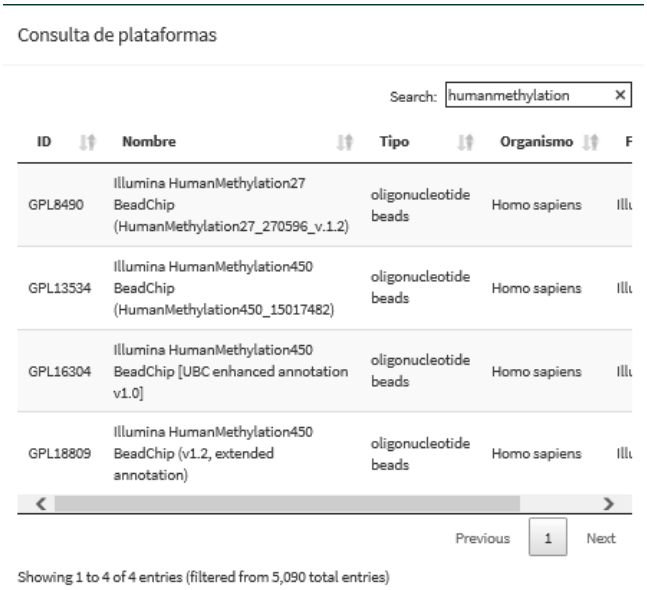


Ilustración 26: Plataformas de un organismo

A continuación se muestra un resumen de las tablas revisadas, así como su situación en la aplicación:

Tabla	Localización (pantalla)	Nombre de enlace
Detalles de plataforma	Seleccionar estudios	Detalles...
Tipos de experimento	Seleccionar estudios	Ayuda...
Detalles de proyecto	Seleccionar estudios	Detalles...
Tipos tumorales	Seleccionar estudios	Tipos...
Valores de las muestras	Analizar muestras	Valores...
Comprobación BD	Configuración	Comprobar GEOmetadb
Consulta de plataformas	Ayuda	Consultar referencias de plataformas GEO. Organismo seleccionado:xxx

2.3 Backend: objetos en memoria y persistencia

En este apartado se describen los diferentes objetos que se van cargando en memoria, a medida que se va avanzando en el proceso de uso de la aplicación, que pasa por las siguientes tareas:

1. Selección de estudios, dentro del rango permitido
2. Selección de muestras, en el rango permitido para las muestras.
3. Análisis de los datos procesados recuperados, o carga manual.
4. Visualización de resultados

A esta lista de tareas habría que añadir una previa que sería la configuración de los parámetros dependiendo de las preferencias del usuario. Seleccionar un límite de muestras menor al que se carga por defecto permitirá un control más preciso del slider, pero si se desea buscar en GDC, habrá que tener en cuenta que cada proyecto se considera un estudio completo, y suelen contener un número de muestras elevados, en especial los grupos con tipos de cáncer más común, como el de mama (BRCA).

No se incluyen en este apartado, las diferentes cadenas que almacenan mensajes de error, categorías, y urls o enlaces.

Objeto	***Descripción***
shiny.maxRequestSize=400* 1024^2	Valor máximo de los ficheros cargados en la pantalla de análisis de muestras
frecAct	Periodo de comprobación de el fichero de base de datos GEOmetadb.sqlite (ms) Establecido en 864000000 (frecuencia diaria)
timeCheckFiles	Tiempo de comprobación de modificación de ficheros de plataformas y proyectos. Establecido en 60000 (1 minuto)
fprojGDC	Almacena los datos recuperados del fichero de plataformas, en caso de fallo de forma online, y en caso de fallo se recupera de la tabla tGDC1
tGDC1, tGDC2	Almacenan información de los proyectos y tipos tumorales respectivamente
fPlfms	Almacena los datos del fichero de plataformas
tOrgan	Almacena el listado de organismos (recuperado de GEO)
defxxx	Valores por defecto de diferentes parámetros que ya se han mencionado anteriormente.

datos$gGSE	Almacena la tabla de resultados devuelta por los distintos filtros involucrados
datos$GSEsel	Almacena sólo los estudios seleccionados del conjunto anterior
datos$GSMsel	Almacena el conjunto de muestras seleccionado
datos$infoGSEs	Almacena información más detallada para mostrar en la pantalla de 'Datos del estudio'
datos$infoGSMs	Almacena la información más detallada de la muestra para mostrar en 'Analizar muestras'
datos$gHist	Almacena el historial de de consultas, que se puede consultar en la pantalla de 'Estadísticas'
datos$GDC	Almacena los datos recuperados por el filtro GDC en el formato nativo del repositorio (sin seleccionar campos)
datos$mensajes	No habilitado, pero mostraría mensajes similares a las notificaciones
datos$notificaciones	Notificaciones para indicar el éxito o aviso de determinadas opciones. Empiezan a mostrarse a partir de la pantalla de 'Datos del estudio'
datos$procesados	Almacena la tabla de datos procesados tal como se muestra en la pantalla de 'Analizar muestras'
datos$valores	Almacena en forma de tabla para cada muestra el tipo de preproceso sufrido. Indica si son beta o M values
datos$methContainer	Almacena los datos procesados en un contenedor RatioSet (minfi). Esto facilita las posibles ampliaciones futuras de la aplicación
datos$anotaciones	Almacena las anotaciones cargadas de la plataforma correspondiente, y es la fuente de la opción Mostrar en la pantalla de 'Analizar muestras'
datosDMR$DMRs	Almacena también en forma de tabla los resultados del análisis DMR con Bumphunter

Hay que aclarar que todos los datos anteriores, al igual que los parámetros de sesión (gXXXXX) son reactivos, y por tanto, los elementos gráficos que los integran se actualizan de forma automática al sufrir modificaciones.

Otros elementos importante que se quedan almacenados en memoria son el estado de los contenedores que contienen las distintas tablas en las distintas pantallas, que se han implementado con la clase DT, entre otras cuestiones, para permitir que algunas de las tablas no permitan

selección , otras tengan selección múltiple y para que almacenen las keywords sobre el campo search de la tabla de resultados de estudios y las distintas selecciones realizadas.

Las distintas tablas implicadas son las siguientes:

ID	***Pantalla***	***Datos del objeto***
d1Stud	Seleccionar estudios	fStu()
d2Smp1...5	Datos del estudio	fSmpls(indice) indice: [1,...,5]
d3Mues	Analizar muestras	datos$GSMsel
d3Proc	Analizar muestras	datos$procesados
d3DMRs	Analizar muestras	datosDMR$DMRs

Queda tratar la parte de persistencia almacenada en forma de ficheros, junto a la base de datos SQLite, que se usa también para consultas. En la siguiente captura de pantalla se pueden observar los ficheros en su ubicación en el sistema operativo.

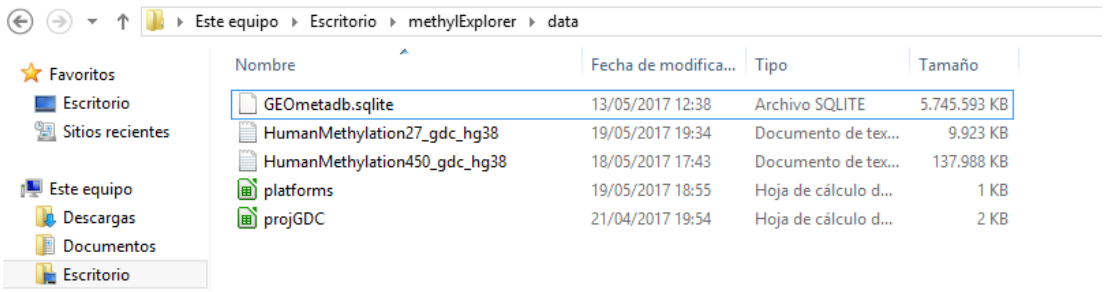


Ilustración 27: Ficheros de persistencia. Carpeta /data

La aplicación tiene capacidad para realizar algunas funciones sin ninguno de los ficheros anteriorse, pero se vería más limitada, ya que no podría acceder a las consultas GEO, y el filtro ArrayExpresss sólo recuperaría los ficheros no redundantes (los no pertenecientes a GEO).

Los ficheros que se adjuntan en la entrega, y que constituyen el funcionamiento normal de la aplicación son por tanto:

GEOmetadb.sqlite: en su versión comprimida ocupa unos 300 Mb., pero la aplicación una vez desplegada lo primero que hace es descargar y descomprimir el fichero en la carpeta /data.

Platforms.csv: almacena la lista de plataformas permitidas. Cada fila representa las equivalentes, o los distintos códigos que puede tener en los distintos repositorios. También sirve para almacenar los paquetes de anotaciones de las plataformas correspondientes.

ProjGDC: almacena la lista de proyectos y su descripción. Se ha decidido de esta forma, porque en un determinado servidor podría decidirse trabajar sólo con un subconjunto de proyectos de GDC, y sólo habría que incluir los ficheros permitidos a modo de lista blanca de seguridad.

HumanMethylation27_gdc_hg38.txt: anotaciones obtenidas de GDC sobre el ensamblado hg38 (Homo sapiens). Para las plataformas permitidas en el fichero entregado, son válidas.

HumanMethylation450_gdc_hg38.txt: análogo al anterior, pero para la plataforma de microarray Illumina Human Methylation 450K.

2.4 Evaluación de resultados

En primer lugar, se va a tratar la gestión de errores realizada, para pasar a continuación a la evaluación de los resultados de los datos de prueba definidos.

Se pueden distinguir dos mecanismos diferentes para mostrar errores, uno son las ventanas de error, que son ventanas de diálogo sin botones y que permiten volver a la aplicación al seleccionar cualquier punto ajeno a la ventana.

En la siguiente captura se observa una de estas ventanas donde ha habido un error en el acceso al repositorio GDC, es decir, el filtro correspondiente no ha podido recuperar datos.

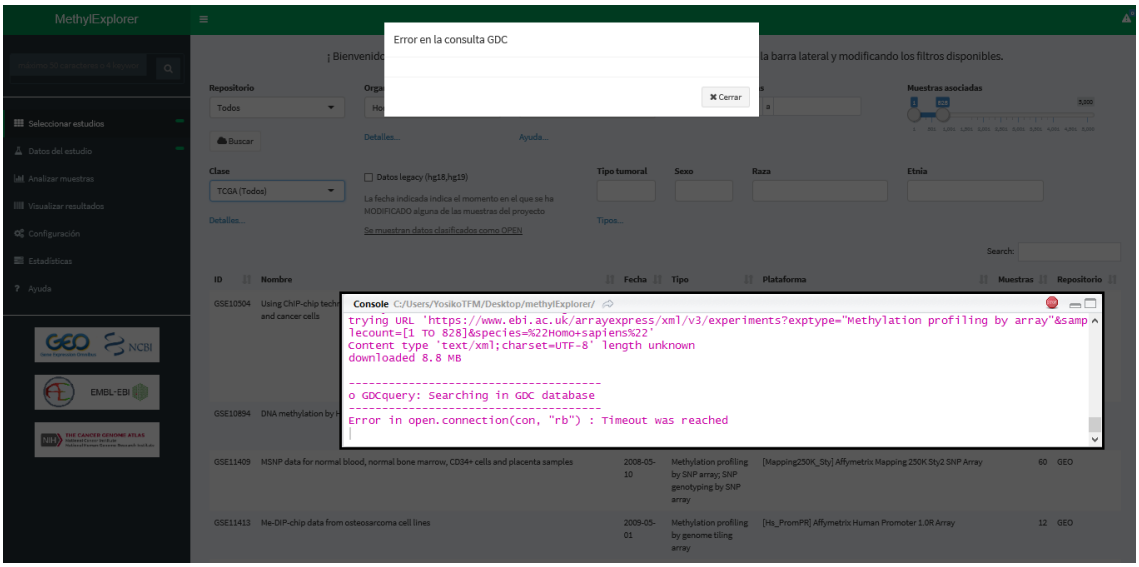


Ilustración 28: Ventanas de error

El otro elemento de información consiste en una serie de notificaciones en la parte derecha del header, y que comienzan a informar de lo que está sucediendo en la aplicación a partir de la pantalla de 'Datos del estudio'. Son varias las notificaciones de tipo 'success' en verde, 'warning' en color naranja y 'danger' en rojo. En los casos en los que la aplicación no responda a los botones seleccionados, habrá que consultar esta parte de la pantalla para ver si se han generado notificaciones nuevas. Para ver si la aplicación está en mitad de un proceso de consulta, adquisición de muestras o análisis hay que observar los botones de la pantalla, ya que los procesos los inhabilitan de forma parcial hasta que se completan las tareas solicitadas.

Como resumen se muestran las distintas notificaciones que puede generar la aplicación

Pantalla	***Tipo***	***Mensaje***	***Significado***

Datos del estudio	Warning	Plataformas no admitidas	Alguna de las plataformas no está admitida
Datos del estudio	Warning	Plataformas no equivlentes	No equivalencia entre al menos un par de muestras
Analizar muestras	Warning	Muestras sin datos RAW disponibles	Al pulsar el botón RAW ninguna de las muestras tiene código 1 o 3
Analizar muestras	Danger	Error con descarga xxx	No se han podido descargar datos de la muestra xxx (botón RAW)
Analizar muestras	Success	Proceso descarga RAW finalizado	Proceso de descarga RAW finalizado (puede haber notificaciones de muestras con error)
Analizar muestras	Warning	Muestra xxx sin datos RAW	Está deshabilitado, ya que aplicaría en caso de habilitar un módulo de análisis de datos RAW
Analizar muestras	Warning	Muestra xxx sin datos procesados	La muestra xxx no tiene datos procesados
Analizar muestras	Warning	Muestra xxx con formato nativo	Se han encontrado muestras en ArrayExpress catalogadas como procesado y en formato nativo
Analizar muestras	Success	Proceso de descarga finalizado	Indica que se ha finalizado con la obtención de muestras (con o sin éxito). Revisar notificaciones previas.
Analizar muestras	Danger	Error con descarga xxx	Error al descargar los datos procesados

En cuanto al sistema de gestión de errores implementado, se basa sobre todo en boloques de tipo tryCatch{...} y tipo try({...},silent=T). El primero se usa en los casos en los que se desea que al generarse el error se realicen determinadas acciones. Un ejemplo está en la llamada a los filtros desde la función principal *fStu*() encargada de generar el dataframe cuyo destino está en la tabla de resultados; el segundo de los operadores se usa en casos en los que no hay incompatibilidades si se finaliza el proceso de forma inesperada.

Otro aspecto relacionado con los errores es la codificación llevada a cabo para validar que los parámetros de las opciones de configuración son correctos. Se muestra un fragmento:

```
...
    } else{
      if ((i<1)||(i>=control$gMaxSp))
        valid <- valid+1
      if (s>defSmpls)
        valid <- valid+10
      if ((c<=1)||(c>5))
        valid <- valid+100
      if ((d<=1)||(d>10))
        valid <- valid +1000 }
```

Es un sistema aditivo posicional binario pero almacenado dentro de un dato de tipo entero. Tras pulsar el botón de confirmar cambios, y dependiendo de las posiciones con valor 1 se puede detectar qué campo es el que no se corresponde a los límites permitidos.

En cuanto a los datos de prueba se definieron los siguientes al inicio del trabajo:

Número	***ID***	***Criterio de selección***
1	GSE46401	Datos procedentes de microarray (plat. GPL13534) y de secuenciado
2	GSE81788	Datos con plataforma GPL8490
3	GSE94356	Datos sólo de secuenciado
4	E-MTAB-5535	Datos con réplicas, de tipo microarray
5	E-ERAD-179	Sólo con datos RAW, con el tipo secuenciado
6	TCGA-LGG	UUID: 001ad307-4ad3-4f1d-b2fc-efc032871c7e

De este subconjunto de datos de prueba, la aplicación en su versión final es capaz de recuperar muestras y mostrar información de todos los estudios, pero al haberse limitado a estudios con microarrays y de las plataformas analizadas, sólo puede analizar datos procesados de muestras de los estudios 1 (las de microarray), 2, 4 y 6.

Este no ha sido el único lote de datos de prueba, aunque se ha probado en su conjunto. Y es que en la definición de requisitos se creía poder afrontar el análisis de todo tipo de estudios y preprocesado de datos, idea que fue modificada con posterioridad ante la limitación de recursos.

2.5 Despliegue de la aplicación

Para realizar un despliegue apropiado de la aplicación, es necesario alojarla en el servidor Shiny Server Open Source, debido a que el servidor shinyapps.io tiene un tamaño máximo de capacidad y genera un error cuando la base de datos SQLite se descomprime, ya que supera los 5 Gb. Actualmente, la aplicación está desplegada en la URL y no tiene un funcionamiento correcto ya que cierra la conexión y se interrumpe el proceso en el despliegue, pasando al estado 'Sleeping'

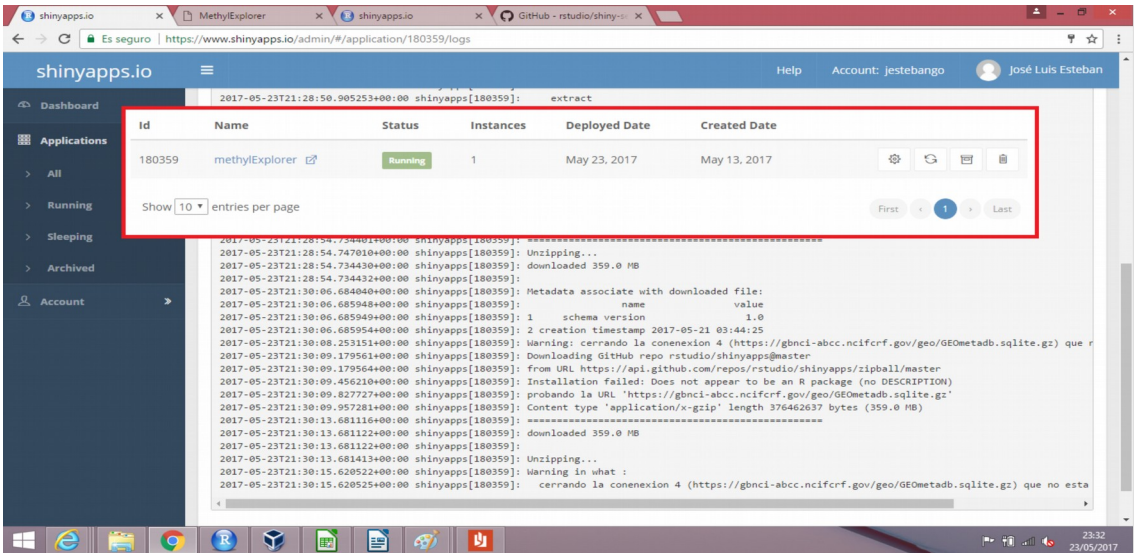


Ilustración 29: Despliegue en shinyapps.io en https://jestebango.shinyapps.io/methylExplorer/

El despliegue se puede realizar con software opensource en un servidor Linux o mediante virtualización. La configuración podría ser la siguiente:

Software de virtualización:	VirtualBox 5.1 (en sistemas Windows)
Sistema operativo:	Ubuntu Server 16.04.2 LTS
Sistema operativo huésped:	Windows 8.1 (IP 192.168.1.23)
Paquetes Linux:	r-base, gdebi-core
Shiny server:	sudo su - -c 'R -e \' install.packages('shiny')\''

La máquina virtual debe tener al menos una conexión de red tipo 'Adaptador puente' para comunicación con el sistema huésped, y con una dirección en el mismo rango (192.168.1.34)

La configuración del servidor Shiny server se realiza sobre el fichero */etc/shiny-server/shiny-server.conf*

Los parámetros para una configuración básica podrían ser los siguientes:

```
log_dir     /var/log/shiny-server   Fichero de logs

server {                            Puerto
 listen 3001 ...

location /methylExplorer {          Ubicación de la aplicación
             app_dir   /srv/shiny-
server/methylExplorer;
  }}
```

La estructura final de ficheros queda como se muestra en la siguiente captura de pantalla:

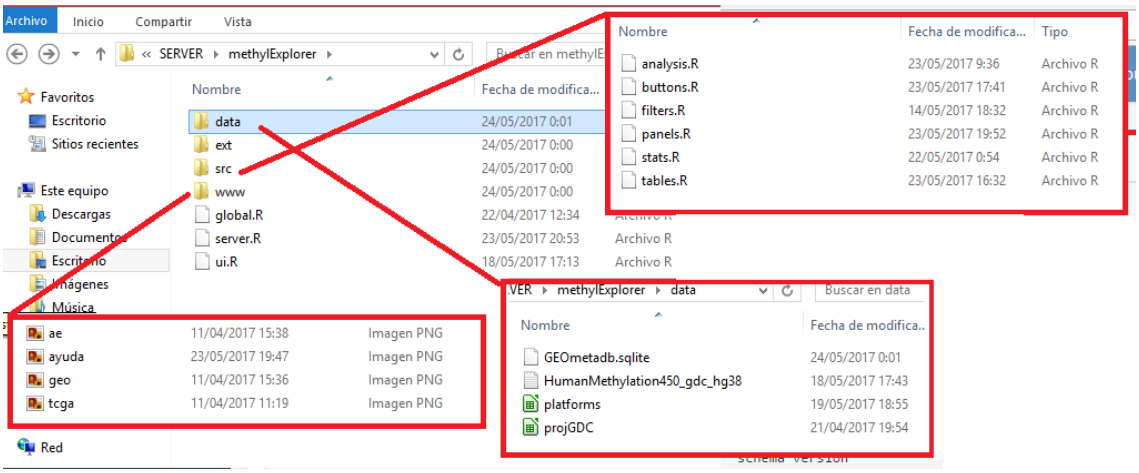


Ilustración 30: Estructura de ficheros de la versión entregable

2.6 Modificaciones respecto la fase de análisis inicial

Una de las principales modificaciones realizadas sobre la aplicación ha sido la inclusión de parámetros configurables, sobre todo para limitar el número de objetos cargados en memoria (factor de rendimiento).

La segunda gran modificación realizada durante las primeras fases fue enfocar la aplicación a una selección de muestras en lugar de una obtención de análisis y estudios completos, minimizando la carga de datos por parte del usuario, ya que la función principal de la aplicación es el modo de exploración.

Otra de las modificaciones importantes, y que afectó a la funcionalidad fue la de selección de datos procesados por no poder abordar la parte de datos sin procesar. También obligó a plantear una solución alternativa permitiendo al usuario la descarga de datos RAW para su uso en aplicaciones externas.

Por otro lado, y como se menciona en uno de los informes de seguimiento, se restó tiempo que estaba destinado en un principio para realizar mejoras, a costa de mejorar los módulos de consulta.

Para finalizar con este apartado, habría que decir que el producto final entregable queda abierto a la inclusión de multitud de mejoras, y está listo para ampliaciones futuras. Respecto al código se ha dividido en varios ficheros y se establece una codificación de elementos de la interfaz que permiten localizar de forma eficiente posibles zonas de código a modificar. Otra mejora que quedó algo incompleta fue la parte de gestión de errores a partir de la pantalla de 'Analizar muestras', donde podrían observarse muestras que no cargan los datos en el formato correcto.

También se han ido realizando cambios con el desarrollo de funciones que se reutilizan en varias zonas de la aplicación, como una de las funciones más usadas parseIn(...).

```
parseIn <- function(pref,cadena,limitadorI,limitadorS,entre,intra)
{
  salida <- ''
  if(length(cadena>0)){

    if (is.null(pref)||pref=='')
      salida <- limitadorI
    else
      salida <- paste(pref,limitadorI)
    for (j in 1:length(cadena)){

      salida <- paste(salida,intra,cadena[j],intra,sep='')
      if (j!=length(cadena))
        salida <- paste(salida,entre,sep='')
    }
```

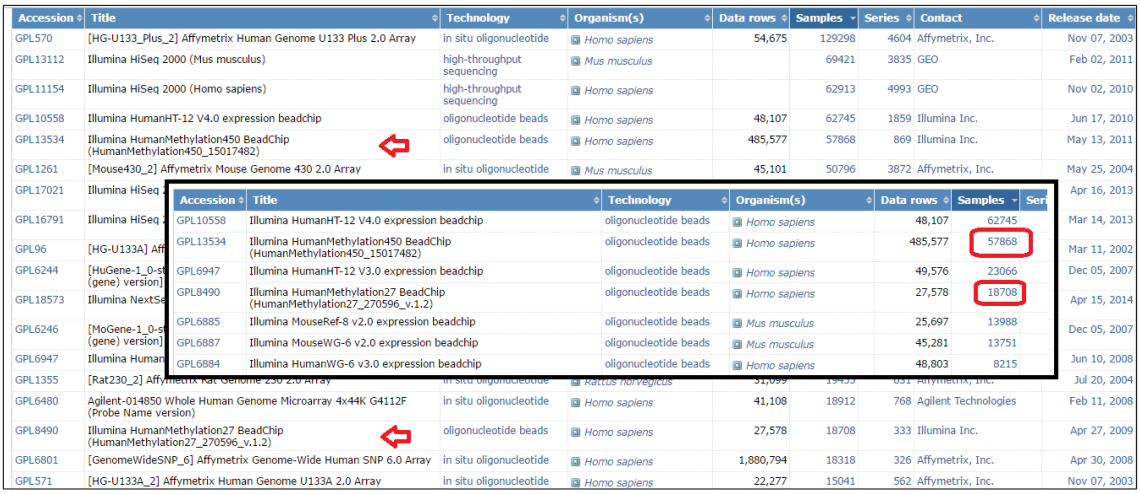
```
    salida <- paste(salida,limitadorS,sep='')
  }
  return(salida)
}
```

Hay funciones que se han ido desarrollando para solucionar problemas concretos, como la función creaURL() que se mencionó en apartados anteriores.

3. Limitaciones y discusión de resultados

No son pocos los retos que se han afrontado para conseguir una de las posibles soluciones a las cuestiones planteadas. Desde el comienzo se ha seguido una metodología para tratar de acotar la heterogeneidad de estudios de metilación que integran los repositorios tratados.

Con este enfoque generalista la idea central del trabajo es facilitar la obtención de datos y la realización de algunos análisis básicos, si bien se ha tenido que dejar de lado muchas cuestiones. Una de las limitaciones que ha sido necesaria es el bloqueo de la aplicación para que recopile y analice datos en función de la plataforma empleada con las muestras concretas, de forma que la aplicación tal como se entrega tiene habilitada funcionalidad con plataformas de tipo microarray, y en concreto, del proveedor Illumina 27K y 450K. No se pueden explorar datos muchos de los datos procedentes de plataformas NGS de secuenciación mediante bisulfito, y cuyos workflows de análisis varían ligeramente al menos en sus fases iniciales, al necesitar un alineamiento previo.



Accession	Title	Technology	Organism(s)	Data rows	Samples	Series	Contact	Release date
GPL570	[HG-U133_Plus_2] Affymetrix Human Genome U133 Plus 2.0 Array	in situ oligonucleotide	*Homo sapiens*	54,675	129298	4604	Affymetrix, Inc.	Nov 07, 2003
GPL13112	Illumina HiSeq 2000 (Mus musculus)	high-throughput sequencing	*Mus musculus*		69421	3835	GEO	Feb 02, 2011
GPL11154	Illumina HiSeq 2000 (Homo sapiens)	high-throughput sequencing	*Homo sapiens*		62913	4993	GEO	Nov 02, 2010
GPL10558	Illumina HumanHT-12 V4.0 expression beadchip	oligonucleotide beads	*Homo sapiens*	48,107	62745	1859	Illumina Inc.	Jun 17, 2010
GPL13534	Illumina HumanMethylation450 BeadChip (HumanMethylation450_15017482)	oligonucleotide beads	*Homo sapiens*	485,577	57868	869	Illumina Inc.	May 13, 2011
GPL1261	[Mouse430_2] Affymetrix Mouse Genome 430 2.0 Array	in situ oligonucleotide	*Mus musculus*	45,101	50796	3872	Affymetrix, Inc.	May 25, 2004
GPL17021	Illumina HiSeq							Apr 16, 2013
GPL16791	Illumina HiSeq							Mar 14, 2013
GPL96	[HG-U133A] Aff							Mar 11, 2002
GPL6244	[HuGene-1_0-st (gene) version]							Dec 05, 2007
GPL18573	Illumina NextSe							Apr 15, 2014
GPL6246	[MoGene-1_0-st (gene) version]							Dec 05, 2007
GPL6947	Illumina Human							Jun 10, 2008
GPL1355	[Rat230_2] Affymetrix Rat Genome 230 2.0 Array	in situ oligonucleotide	*Rattus norvegicus*	31,099	19455	651	Affymetrix, Inc.	Jul 20, 2004
GPL6480	Agilent-014850 Whole Human Genome Microarray 4x44K G4112F (Probe Name version)	in situ oligonucleotide	*Homo sapiens*	41,108	18912	768	Agilent Technologies	Feb 11, 2008
GPL8490	Illumina HumanMethylation27 BeadChip (HumanMethylation27_270596_v.1.2)	oligonucleotide beads	*Homo sapiens*	27,578	18708	333	Illumina Inc.	Apr 27, 2009
GPL6801	[GenomeWideSNP_6] Affymetrix Genome-Wide Human SNP 6.0 Array	in situ oligonucleotide	*Homo sapiens*	1,880,794	18318	326	Affymetrix, Inc.	Apr 30, 2008
GPL571	[HG-U133A_2] Affymetrix Human Genome U133A 2.0 Array	in situ oligonucleotide	*Homo sapiens*	22,277	15041	562	Affymetrix, Inc.	Nov 07, 2003

Accession	Title	Technology	Organism(s)	Data rows	Samples	Ser
GPL10558	Illumina HumanHT-12 V4.0 expression beadchip	oligonucleotide beads	*Homo sapiens*	48,107	62745	
GPL13534	Illumina HumanMethylation450 BeadChip (HumanMethylation450_15017482)	oligonucleotide beads	*Homo sapiens*	485,577	57868	
GPL6947	Illumina HumanHT-12 V3.0 expression beadchip	oligonucleotide beads	*Homo sapiens*	49,576	23066	
GPL8490	Illumina HumanMethylation27 BeadChip (HumanMethylation27_270596_v.1.2)	oligonucleotide beads	*Homo sapiens*	27,578	18708	
GPL6885	Illumina MouseRef-8 v2.0 expression beadchip	oligonucleotide beads	*Mus musculus*	25,697	13988	
GPL6887	Illumina MouseWG-6 v2.0 expression beadchip	oligonucleotide beads	*Mus musculus*	45,281	13751	
GPL6884	Illumina HumanWG-6 v3.0 expression beadchip	oligonucleotide beads	*Homo sapiens*	48,803	8215	

Ilustración 31: Captura realizada en el buscador de plataformas en GEO

Los criterios para elegir estas plataformas se pueden apreciar en la imagen, en la que se ordena por número de muestras comparando con el resto de plataformas, y con arrays de Illumina.

La segunda de las limitaciones importantes se ha producido por motivos temporales y de capacidad de procesamiento, y ha sido el hecho de elegir entre la obtención y tratamiento de los datos en bruto, o comenzar a partir de los datos procesados. Comenzar con los datos RAW tiene innumerables ventajas, ya que supone validar el preprocesamiento y controles de calidad para llegar a unos resultados con menor sesgo. La desventaja y el motivo de haberlos descartado es el gran volumen que suponen tanto en su descarga (requiere gran ancho de banda al menos

para las pruebas) y la capacidad de procesamiento mayor. No obstante, a la aplicación se le ha dotado de funcionalidad para distinguir los tipos de datos de cada muestra de los repositorios, ya que es muy común encontrar estudios con sólo datos de tipo RAW, sólo con datos procesados y estudios con ambos tipos.

Otra de las limitaciones que se aprecian al analizar la funcionalidad la aplicación es la elección en los tipos de análisis y gráficos para visualizar los datos procesados obtenidos de las muestras. Se echan en falta muchas de las gráficas que ayudan a visualizar hipo e hipermetilación sobre todo a nivel de sonda o posición CpG concreta: heatmap, gráficas lollipop o pistas en contexto cromosómico son otras de las carencias que se han tenido en mente en todo momento y que no se han podido afrontar. Respecto a los análisis, se podrían haber otros algoritmos para encontrar DMR o la capacidad de modificar los parámetros en el que se ha seleccionado (BumpHunter); buscando un equilibrio entre eficiencia y capacidad computacional se realiza de forma fija con 10 permutaciones y un cutoff de 0.2 en el modelo.

También han quedado pendientes mejoras que mejorarían la velocidad de consulta con los repositorios, en concreto, con GDC. La estrategia usada es la API implementada en TCGAbiolinks con GDCquery, pero recientemente se ha publicado el paquete GenomicDataCommons en Bioconductor que parece dar solución a uno de los problemas encontrados, que es el cuello de botella formado al realizar consultas a varios proyectos simultáneos (de ahí el diseño en forma de selección simple, con una opción de TCGA-Todos); si al menos uno de los proyectos consultados no satisface la consulta no se devuelven los resultados del resto (un ejemplo es que no contenga todavía datos de metilación publicados, como la mayoría de proyectos TARGET en la fecha de elaboración de este documento).

En relación a las distintas estrategias de filtrado y consulta también existen alternativas, que pueden suponer menos problemas en algunos casos. Seríe el caso del repositorio GEO en el que se ha elegido un sistema de consultas a una base de datos local que es necesario descargar de forma periódica. La ventaja es una reducción del tiempo de respuesta y adquisición de datos, pero el coste es la dependencia de que se actualice la base de datos y que hay una ventana de tiempo hasta la actualización que puede ser de unos 15 días aproximadamente, y que puede automatizarse desde GEOmetadb. De hecho, la aplicación tiene la función de una descarga periódica diaria que se puede configurar en el servidor (en el código, ya que la base de datos es única a todas las sesiones y no sería razonable que cualquier usuario pudiera modificarla).

Otra capacidad importante de la aplicación es la de adquirir muestras individuales de diferentes estudios/repositorios y combinarlas entre sí en una especie de metaanálisis, lo que introduce sesgos adicionales al tratar con datos procesados (beta values o M values) tratados con

software diferente, además de los tratamientos diferenciales a los que se han sometido las muestras antes de ser escaneadas en las diversas plataformas. La contraprestación es la obtención de fenotipos más diversos, y que a veces, es difícil reunir en un mismo estudio, Otra función importante de la aplicación, al menos se ha considerado así, es la capacidad de aislar las muestras de los distintos estudios, de forma que si en un mismo estudio hay muestras de distintas plataformas, hay capacidad de discriminar entre ellas y permitir la adquisición y análisis posterior de las que se consideran válidas, en el fichero de datos.

La reutilización de código tiene su máximo exponente en este trabajo en la función parseIn (en server.R). Originada para unir un array de caracteres con un formato determinado, en concreto, la salida IN ('…','…','…','…') original, se fue modificando de tal forma que se ha usado durante casi todos los módulos en sus diferentes combinaciones: sin prefijo, sin paréntesis, sin comillas, etc.

Por último, sería conveniente destacar el sistema de marcado de estudios diseñado para tal efecto mediante un flag (A,S,AS). Se ha ideado como sistema de validación a las categorías usadas por cada repositorio. En este sentido se han detectado estudios como el GSE62929 catalogados como perfiles de secuenciado, pero en los que aparece como única plataforma GPL13534 que corresponde a un array de metilación Illumina. También habría que matizar ya que los estudios validados (curated) son los que tienen además prefijo GDSs. y pueden englobar a distintos GSEs.

4. Conclusiones

Como punto de partida, destaca el hecho de realizar una correcta planificación y tener en cuenta posibles riesgos y cómo mitigarlos en caso de materializarse. En mi caso, uno de los más temidos ha sido subestimar algunos requisitos planteados desde el comienzo, y que me han obligado a limitar la funcionalidad de la aplicación, aunque en todo caso he primado la calidad sobre el volumen de código implementado.

Esta subestimación se ha concretado en los siguientes puntos:

- Estimación incorrecta de la dificultad de adquirir determinados datos de los repositorios. Aunque por lo general están bastante bien estructurados, la información es bastante heterogénea de forma general. Un ejemplo son los datos procesados de algunos estudios de ArrayExpress, en los que puede existir un único fichero en distintos formatos con columnas diferentes para cada muestra, o ficheros individualizados también con diferentes formatos. Este ha sido el principal motivo de introducir un apartado de importación de datos, para permitir una carga de datos externa mediante fichero, en los casos en los que sea compleja la obtención de forma automática.

- Exceso de confianza con algunos paquetes, por ejemplo con GEOquery. En apariencia es una función que se desenvuelve bien en la estructura de directorios del repositorio GEO, con funciones concretas para adquirir datos suplementarios de muchos estudios; pero el uso continuado a lo largo del proyecto ha demostrado su ineficiencia en algunos casos, de forma que pueden ser menos problemáticas las adquisiciones FTP (o HTTP con download.file()) automatizadas.

- Falta de previsión en la adquisición de datos. Esto se ha manifestado en multitud de ocasiones, y sobre todo durante las pruebas modulares de adquisición de los datos procesados. En este sentido se ha mejorado la capacidad de anticipación ante «cisnes negros». Se pueden citar varios: no prever que algunos datos sumarizados pueden ser M Values en lugar de betas, creencia inadecuada de que se puede gestionar cualquier volumen de datos (algunos han supuesto excesivos problemas, sobre todo con microarrays más densos como el 450K).

- Subestimar las características fenotípicas de las muestras. Ante la creencia inicial de que sería relativamente sencillo desglosar los fenotipos de las distintas muestras adquiridas, ya que en algunos casos se puede importar

directamente a un objeto tipo *ExpressionSet* (y con *pData*() o *colNames*() se adquieren). La realidad es que aunque pueden desglosarse «en origen», el problema viene después al combinarlos ya que utilizan diferentes nomenclaturas y habría que crear un sistema de diccionario de equivalencias (similar al implementado con las plataformas), para aunarlos a todos; ej: *tissue, characteristic_type*,…). La solución implementada puede parecer limitada tomando el fenotipo como un resumen de sus características / factores, de ahí que se haya implementado el sistema de inclusión manual de grupos para realizar el análisis DMR. Teniendo esta carencia en mente se permite al usuario navegar por las distintas pantallas para revisar las muestras que forman parte del estudio y las seleccionadas, y se ha implementado un sistema de información en el sidebar para que se visualice en todo momento el número de estudios y muestras cargados en memoria.

Otra de las constantes durante el trabajo es la búsqueda de un equilibrio entre eficiencia y rendimiento. A medida que se han ido desarrollando las pruebas, se ha hecho cada vez más necesaria la implementación de un sistema que limite el número de objetos cargados en memoria, debido a los recursos limitados implicados en el trabajo. Es el motivo de la creación de los parámetros que limitan el máximo de estudios/muestras y que se han implementado como configurables de sesión (para cada usuario). De esta forma, en equipos con gran capacidad computacional, la aplicación se podría configurar para aumentar estos límites.

Otro de los retos acaecidos y que ha supuesto un desafío continuo es evitar caer en la distracción de la mejora continua. En todas las fases del proyecto, se han ido plasmando mejoras, muchas de las cuales ha sido inviable realizar, pero que podrían haber provocado una gran desviación si no se hubiera realizado la planificación inicial. En las primeras fases es donde se puede comprobar mejor este hecho, y es la razón de que el módulo de importación haya sido el que ha sufrido un mayor número de mejoras.

Respecto a la consecución de logros, a pesar del enorme tiempo y esfuerzo invertido en el trabajo, no me siento completamente satisfecho, ya que a pesar de haberlo ido acotando sigo ideando futuras mejoras así como la inclusión de nuevas funcionalidades que estoy revisando sobre todo en paquetes Bioconductor nuevos que se van publicando, como *shinyMethyl.*

Otra de las habilidades adquiridas durante el desarrollo ha sido valorar la definición de unos requisitos lo más concisos posible para cumplir con los objetivos. Es bastante complejo sobre todo al abordar pipelines o tecnologías por primera vez, pero a la vez es necesario para ir

evolucionando a medida que se van diseñando perspectivas y distintos enfoques.

La metodología usada en este trabajo de diseño modular no ha sido una novedad por tener experiencia en la asignatura Prácticas de Empresa con esta tecnología, pero sí lo han sido algunas de las estrategias que se han ido diseñando ante la magnitud inicial del problema. Los cambios en cuanto a diseño han sido varios y se han comentado ya en apartados anteriores.

Hay que decir que las líneas de trabajo futuro que puedan tomar como base este trabajo irían enfocadas sobre todo en cubrir otros pipelines de análisis de metilación no tratados, así como la adquisición de datos procedentes de otros repositorios, ya que se han contemplado los que albergan mayor número de muestras. Al ser una aplicación web con cierta autonomía, en el sentido de que depende fundamentalmente de la comunicación con los repositorios, tampoco se ha centrado en el análisis de un pipeline concreto o el desarrollo de nuevos modelos que aumenten la precisión en la detección de marcadores o artefactos tipo CpG.

La aplicación por tanto, se considera mejorable en todos los aspectos, desde la optimización de los filtros, a la inclusión de gráficas más completas. Por otro lado, lo que se ha logrado cumplir con éxito, al menos en mi opinión, es la funcionalidad que le da nombre, y es la de exploración y capacidad de personalizar las consultas. El número criterios de búsqueda y filtros también se puede aumentar de forma considerable, introduciendo búsquedas por fenotipos de las muestras por ejemplo, aunque considero que con los recursos que se han usado en el desarrollo, se ha obtenido una interfaz aceptable, buscando como se ha mencionado anteriormente siempre el equilibrio entre capacidad y rendimiento.

5. Glosario

API	Application Programming Interface
DMR	Differentially Methylated Regions
DNA	Desoxyribonucleic acid
DNMTs	DNA methyltransferase enzymes
GDC	Genomic Data Commons
GEO	Gene Expression Omnibus
kb	kilobases o miles de bases (referido al DNA)
HPLC	High Performance Liquid Chromatography
IDF	Investigation Description Format
LncRNA	long non-coding RNA
LUMA	LUminometric Methylation Assay
miRNA	micro RNA
MAGE-TAB	Microarray Gene Expression Tabular Format
piRNA	Piwi RNA (asociado a la família de proteínas Piwi)
PCR	Polymerase Chain Reaction
RAW	Relativo a los datos, se refiere a los obtenidos de la plataforma y sin procesar
RNA	Ribonucleic acid
RRBS	Reduced Representation Bisulfite Sequencing
SDRF	Sample and Data Relationship Format
siRNA	small inhibitory RNA
sncRNA	small non-coding RNA
SNP	Single Nucleotide Polymorphism
SQLi	SQL injection es un tipo de ataque a SQL
TCGA	The Cancer Genome Atlas
UUID	Universally Unique Identifier
WGBS	Whole Genome Bisulfite Sequencing

6. Bibliografía

1. Kurdyukov, S., & Bullock, M. (2016). DNA Methylation Analysis: Choosing the Right Method. Biology, 5(1), 3. http://doi.org/10.3390/biology5010003

2. Du, P., Zhang, X., Huang, C.-C., Jafari, N., Kibbe, W. A., Hou, L., & Lin, S. M. (2010). Comparison of Beta-value and M-value methods for quantifying methylation levels by microarray analysis. *BMC Bioinformatics*,*11*, 587. http://doi.org/10.1186/1471-2105-11-587

3. Barrett T, Wilhite SE, Ledoux P, Evangelista C, Kim IF, Tomashevsky M, Marshall KA, Phillippy KH, Sherman PM, Holko M, Yefanov A, Lee H, Zhang N, Robertson CL, Serova N, Davis S, Soboleva A.
NCBI GEO: archive for functional genomics data sets--update.
Nucleic Acids Res. 2013 Jan;41(Database issue):D991-5.

4. Kolesnikov N. et al., 2015.ArrayExpress update-simplifying data submissions.. Nucleic Acids Res, doi:10.1093/nar/gku1057. Pubmed ID 25361974.

5. Grossman, Robert L., Heath, Allison P., Ferretti, Vincent, Varmus, Harold E., Lowy, Douglas R., Kibbe, Warren A., Staudt, Louis M. (2016) Toward a Shared Vision for Cancer Genomic Data. New England Journal of Medicine 375:12, 1109-1112

6. *Shiny Server v1.4.0 Configuration Reference*. (2017) .*Rstudio.github.io*. Retrieved 1 May 2017, from http://rstudio.github.io/shiny-server/os/latest/

7. Beeley, C. (2016).*Web application development with R using Shiny* (1st ed.). Birmingham: Packt.

8. Antonio Colaprico, Tiago Chedraoui Silva, Catharina Olsen, Luciano Garofano, Claudia Cava, Davide Garolini, Thais Sabedot, Tathiane Malta, Stefano M. Pagnotta, Isabella Castiglioni,Michele Ceccarelli, Gianluca Bontempi Houtan Noushmehr. *TCGAbiolinks: An R/Bioconductor package for integrative analysis of TCGA data.* Nucleic Acids Research (05 May 2016) 44 (8): e71. (doi:10.1093/nar/gkv1507)

9. Makhabel, B.*Learning data mining with R* (1st ed.)

10. Jaffe, A. E., Murakami, P., Lee, H., Leek, J. T., Fallin, M. D., Feinberg, A. P., & Irizarry, R. A. (2012). *Bump hunting to identify differentially methylated regions in epigenetic epidemiology studies*. International Journal of Epidemiology,*41*(1), 200–209. http://doi.org/10.1093/ije/dyr238

11. Zhu Y, Davis S, Stephens R, Meltzer PS, Chen Y. *GEOmetadb: powerful alternative search engine for the Gene Expression Omnibus*. Bioinformatics. 2008 Dec 1;24(23):2798-800. doi:

10.1093/bioinformatics/btn520. Epub 2008 Oct 7. PubMed PMID: 18842599; PubMed Central PMCID: PMC2639278.

12. Martin Morgan and Davis Sean (2017). *GenomicDataCommons*: NIH / NCI Genomic Data Commons Access. R package version 1.0.0.

13. Aryee MJ, Jaffe AE, Corrada-Bravo H, Ladd-Acosta C, Feinberg AP, Hansen KD and Irizarry RA (2014). "*Minfi: A flexible and comprehensive Bioconductor package for the analysis of Infinium DNA Methylation microarrays*." Bioinformatics_, *30*(10), pp. 1363-1369. doi:10.1093/bioinformatics/btu049URL: http://doi.org/10.1093/bioinformatics/btu049).

14. Kauffmann, A., Rayner, T.F., Parkinson, H., Kapushesky, M., Lukk, M., Brazma, A., Huber, W. (2009) *Importing ArrayExpress datasets* into R/Bioconductor. Bioinformatics, 25(16):2092-4

15. Views, R. (2017).*Databases using R.Rviews.rstudio.com*. Retrieved 1 May 2017, from https://rviews.rstudio.com/2017/05/17/databases-using-r/

16. *Shiny - Scoping rules for Shiny apps*. (2017).*Shiny.rstudio.com*. Retrieved1 May 2017, from https://shiny.rstudio.com/articles/scoping.html

7. Anexos

7.1 Código fuente

```
#-------------------------------------------------------------------------------
# Global.R
#-------------------------------------------------------------------------------

  ventanaError <- function(msj){
    advertencia <-
modalDialog(footer=modalButton('Cerrar',icon=shiny::icon('close')),title=msj,size = 'm')
    showModal(advertencia)
  }
```

```
#-------------------------------------------------------------------------------
# ui.R
#-------------------------------------------------------------------------------

library(shiny)
library(shinydashboard)
library(shinyjs)

# Publicada en https://jestebango.shinyapps.io/methylExplorer

# User-interface

urlGEO <- 'https://www.ncbi.nlm.nih.gov/gds'
urlAE <- 'https://www.ebi.ac.uk/arrayexpress/browse.html'
urlTCGA <- 'https://portal.gdc.cancer.gov/'

#-----------------+-----------------------------------------------


shinyUI <-  dashboardPage(skin='green',title='MethylExplorer',


  #HEADER
  dashboardHeader(title ='MethylExplorer',titleWidth = 300,
    # dropdownMenuOutput('msjMenu'),
    dropdownMenuOutput('notMenu')),

  #SIDEBAR
  dashboardSidebar(width=300,

    tags$br(),
    sidebarSearchForm(textId='kw',label='máximo 50 caracteres o 4 keywords',buttonId='b0Selt'),
    tags$hr(),

    sidebarMenu(id='menu',
                              menuItem('Seleccionar     estudios',     icon     =     icon('th'),
tabName='opcion1',badgeLabel=textOutput('sbEsts')),
                            menuItem('Datos     del     estudio',     icon     =     icon('flask'),
tabName='opcion2',badgeLabel=textOutput('sbSmps')),
      menuItem('Analizar muestras', icon = icon('bar-chart-o'), tabName='opcion3'),
      menuItem('Visualizar resultados', icon = icon('barcode'), tabName='opcion4'),
      menuItem('Configuración', icon = icon('gears'), tabName='opcion5'),
      menuItem('Estadísticas', icon = icon('tasks'), tabName='opcion6'),
      menuItem('Ayuda', icon = icon('question'), tabName='opcion7')
    ),

     tags$hr(),tags$div(id='logos',align='center',tags$a(href=urlGEO,target="_blank",tags$img(sr
c='geo.png',border=1))),tags$br(),
       tags$div(id='logo2',align='center',tags$a(href=urlAE,target="_blank",tags$img(src='ae.png
',border=1))),tags$br(),
       tags$div(id='logo3',align='center',tags$a(href=urlTCGA,target="_blank",tags$img(src='tcga
.png',border=1)))

  ),
```

```
  #BODY
  dashboardBody(
    useShinyjs(),
    uiOutput('UIMain')
  )
)
```

```
#--------------------------------------------------------------------------------
# server.R
#--------------------------------------------------------------------------------

# source('https://bioconductor.org/biocLite.R')
library(GEOquery)
library(GenomicDataCommons)
library(GEOmetadb)
library(ArrayExpress)
library(TCGAbiolinks)
library(minfi)
library(Gviz)
# library(IlluminaHumanMethylation27k.db)
# library(IlluminaHumanMethylation450k.db)
library(shiny)
library(shinyjs)
library(shinydashboard)
library(gtools)
library(DT)
library(devtools)
library(ggplot2)
library(plotly)
library(gdata)
library(scales)
library(stringi)
library(stringr)
library(plotrix)
library(RSQLite)
library(dplyr)
library(tidyr)
library(tm)
library(wordcloud)
library(XML)

# Opciones globales

options(shiny.maxRequestSize=400*1024^2)
# rsconnect::configureApp("methylExplorer", size="xxxlarge")

#-------------------------------------------------------------------------------

# Actualización de GEOmetadb según la frecuencia especificada

# frecAct <- 300000 # 10 minutos (pruebas)
# ¡¡ Quitar comentarios tras pruebas !!

# msjeAct <- 'Se está actualizando GEOmetadb.rsqlite'
#
#
frecAct <- 86400000 #(frecuencia diaria)

observe({
  getSQLiteFile(destdir='data')
  invalidateLater(frecAct,session)
})

#
#   # Actualización GEO SQLite
#   # Condiciones de actualización:
#   #   1. No se ha descargado nunca
#   #   2. Ha transcurrido una semana (hacerlo configurable)
#
#   updateGEO <- function(){
#
#     cDate <- file.info('data/GEOmetadb.sqlite')$ctime
```

```
#     tDate <- Sys.time()
#     if ((difftime(tDate,cDate,units='weeks')>1)||(file.exists('data/GEOmetadb.sqlite')))
#       getSQLiteFile('data')
#   }

#-------------------------------------------------------------------------------

# outputOptions(output, 'download', suspendWhenHidden=FALSE)

# Tiempo de comprobación para carga dinámica de ficheros

timeCheckFiles <- 60000 # 1 minuto

# URL para queries XML a AE

urlAE <- c('https://www.ebi.ac.uk/arrayexpress/xml/v3/experiments?',
           'https://www.ebi.ac.uk/arrayexpress/jason/v3/experiments?')

qAEGen <- 'https://www.ebi.ac.uk/arrayexpress/xml/v3/experiments?exptype=
%22Methylation+profiling+by+array%22+OR+%22Methylation+profiling+by+high+throughput+sequencing
%22+OR+%22Bisulfite-seq%22+OR+%22MeDIP-seq%22&species=%22homo+sapiens%22'

# URL para consultar más información de los estudios seleccionados

urlConGEO <- 'https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc='
urlConAE <- 'http://www.ebi.ac.uk/arrayexpress/experiments/'
urlConGDC <- 'https://portal.gdc.cancer.gov/projects/'

#-------------------------------------------------------------------------------
# FUNCIONES

# Mover archivos a otra carpeta

moveFile <- function(from, to) {
  todir <- dirname(to)
  if (!isTRUE(file.info(todir)$isdir)) dir.create(todir, recursive=TRUE)
  file.rename(from = from,  to = to)
}

#-------------------------------------------------------------------------------

# Extrae los últimos n caracteres del string x

substrRight <- function(x, n){
  substr(x, nchar(x)-n+1, nchar(x))
}

#-------------------------------------------------------------------------------


# Query GEO

fGEO <- function(sql){

  if(!file.exists('data/GEOmetadb.sqlite'))
    getSQLiteFile(destdir='data')
  con <- dbConnect(SQLite(),'data/GEOmetadb.sqlite')
  info <- dbGetQuery(con,sql)
  dbDisconnect(con)
  return(info)
}

#-------------------------------------------------------------------------------

# Cambio a formato de cto de cadenas prefijo  ('...','...',...) para consultas SQLite

parseIn <- function(pref,cadena,limitadorI,limitadorS,entre,intra)
{
  salida <- ''
  if(length(cadena>0)){

    if (is.null(pref)||pref=='')
      salida <- limitadorI
    else
```

```
      salida <- paste(pref,limitadorI)
    for (j in 1:length(cadena)){

      salida <- paste(salida,intra,cadena[j],intra,sep='')
      if (j!=length(cadena))
        salida <- paste(salida,entre,sep='')
    }
    salida <- paste(salida,limitadorS,sep='')
  }
  return(salida)
}

#-------------------------------------------------------------------------------------

# Separa en palabras una cadena sólo si la longitud <= 50 caracteres y <= 4 palabras
# Puede devolver ''(no conversión) o la cadena troceada

parseSep <- function(sinSep){

  separada <- ''
  size <- stri_length(sinSep)
  if ((size>0)&&(size<=50))
  {
    sinSep <- removePunctuation(sinSep)
    words <- strsplit(sinSep,' ')[[1]]
    #words <- str_to_upper(words)
    words <- unique(words)
    if(length(words)<=4)
      separada <- words
  }

  return(separada)
}

#-------------------------------------------------------------------------------------

# Parse keywords de búsqueda. Limitaciones: 4 keywords o 50 caracteres
# Salida: 'field LIKE '%...%' OR/AND field LIKE '%...%'

parseLike <- function(field,cadena,operador){

  qu <- ''
  if (operador!='')
    words <- parseSep(cadena)
  else
    words <- cadena

  for (i in 1:length(words))
  {
    qu <- paste(qu,field," LIKE '%",words[i],"%'",sep='')
    if (i!=length(words))
      qu <- paste(qu,' ',operador,' ',sep='')

  }
  return(qu)
}

shinyServer(function(input, output,session) {

#-------------------------------------------------------------------------------------
# VARIABLES

  # Mensaje en pantalla de inicio

  m1msjI <- '¡ Bienvenido a MethylExplorer ! Puede realizar búsquedas mediante el formulario
situado en la barra lateral y modificando los filtros disponibles.'

  catDisID <- c('TCGA-SARC','TCGA-KIRP','TCGA-PAAD','TCGA-READ','TARGET-NBL','TCGA-GBM','TCGA-
MESO','TCGA-ACC','TARGET-OS','TCGA-BRCA',
                'TCGA-CESC','TARGET-RT','TCGA-ESCA','TCGA-KICH','TCGA-DLBC','TCGA-KIRC','TCGA-
SKCM','TCGA-LAML','TCGA-PCPG',
                'TARGET-AML','TCGA-UVM','TCGA-LUSC','TCGA-UCS','TCGA-COAD','TCGA-TGCT','TCGA-
HNSC','TCGA-UCEC','TCGA-BLCA','TARGET-CCSK',
                'TCGA-LIHC','TCGA-LGG','TCGA-THCA','TCGA-PRAD','TARGET-WT','TCGA-OV','TCGA-
```

```
CHOL','TCGA-STAD','TCGA-THYM','TCGA-LUAD')

  catDisDs <- c('Sarcoma','Kidney Renal Papillary Cell Carcinoma','Pancreatic Adenocarcinoma',
                'Rectum Adenocarcinoma','Neuroblastoma','Glioblastoma
Multiforme','Mesothelioma',
                'Adrenocortical Carcinoma','Osteosarcoma','Breast Invasive Carcinoma',
                'Cervical Squamous Cell Carcinoma and Endocervical Adenocarcinoma',
                'Rhabdoid Tumor','Esophageal Carcinoma','Kidney Chromophobe',
                'Lymphoid Neoplasm Diffuse Large B-cell Lymphoma',
                'Kidney Renal Clear Cell Carcinoma','Skin Cutaneous Melanoma',
                'Acute Myeloid Leukemia (PS: Bone Marrow)','Pheochromocytoma and
Paraganglioma',
                'Acute Myeloid Leukemia (PS: Blood)','Uveal Melanoma','Lung Squamous Cell
Carcinoma',
                'Uterine Carcinosarcoma','Colon Adenocarcinoma','Testicular Germ Cell Tumors',
                'Head and Neck Squamous Cell Carcinoma','Uterine Corpus Endometrial Carcinoma',
                'Bladder Urothelial Carcinoma','Clear Cell Sarcoma of the Kidney',
                'Liver Hepatocellular Carcinoma','Brain Lower Grade Glioma',
                'Thyroid Carcinoma','Prostate Adenocarcinoma','High-Risk Wilms Tumor',
                'Ovarian Serous Cystadenocarcinoma','Cholangiocarcinoma',
                'Stomach Adenocarcinoma','Thymoma','Lung Adenocarcinoma')

  catPS <- c('Soft tissue','Kidney','Pancreas','Colorectal','Nervous
System','Brain','Pleura','Adrenal Gland','Bone','Breast',
             'Cervix','Kidney','Esophagus','Kidney','Lymph Nodes','Kidney','Skin','Bone
Marrow','Adrenal Gland','Blood','Eye',
             'Lung','Uterus','Colorectal','Testis','Head and
Neck','Uterus','Bladder','Kidney','Liver','Brain','Thyroid','Prostate',
             'Kidney','Ovary','Bile Duct','Stomach','Thymus','Lung')

  tGDC1 <- data.frame(catDisID,catDisDs,catPS)
  colnames(tGDC1) <- c('ID','Clase','PS')

  catSmpID <-
c('TP','TR','TB','TRBM','TAP','TM','TAM','THOC','TBM','NB','NT','NBC','NEBV','NBM','CELLC','TRB
','CELL','XP','XCL')

  catSmpDs <- c('Primary solid Tumor','Recurrent Solid Tumor','Primary Blood Derived Cancer -
Peripheral Blood',
                'Recurrent Blood Derived Cancer - Bone Marrow','Additional - New
Primary','Metastatic','Additional Metastatic',
                'Human Tumor Original Cells','Primary Blood Derived Cancer - Bone
Marrow','Blood Derived Normal',
                'Solid Tissue Normal','Buccal Cell Normal','EBV Immortalized Normal','Bone
Marrow Normal','Control Analyte',
                'Recurrent Blood Derived Cancer - Peripheral Blood','Cell Lines','Primary
Xenograft Tissue','Cell Line Derived Xenograft Tissue')

  tGDC2 <- data.frame(catSmpID,catSmpDs)
  colnames(tGDC2) <- c('ID','Tipo')

  catGendGDC <- c('Male','Female','Not Reported','Unknown')
  catRaceGDC <- c('Black Or African American','Asian','White','American Indian Or Alaska
Native','Native Hawaiian Or Other Pacific Islander','Other','Not Reported')
  catEthnGDC <- c('Hispanic Or Latino','Not Hispanic Or Latino','Not Reported')

  #---------------------------------------------------------------------------------

  # Carga de los datos de las plataformas aceptadas en la aplicación y de los proyectos GDC
(Prioridad: 1.Fichero, 2.API, 3.Memoria)
  # Plataformas de búsqueda por defecto (si no se especifican en fichero): GPL1,GPL2

  defGPL <- data.frame(c('GPL13534'),c('Illumina MethylationHuman450K'),c('A'))
  colnames(defGPL) <- c('ID','Nombre','Tipo')

  fprojGDC <- tGDC1

  fPlfms <- as.data.frame(read.csv('data/platforms.csv',stringsAsFactors = F,header=T))
  if (!exists('fPlfms'))
  fPlfms <- defGPL

  #---------------------------------------------------------------------------------

  # Actualización dinámica del fichero de los ficheros de plataformas y proyectos
```

```
  # Mejora portencial: distinta frecuencia de "observación" (proyectos no debería cambiar
mucho)

  observe({
    invalidateLater(timeCheckFiles, session)


    try({
      if (exists(fPlfms) && (exists(fprojGDC)))
        rm(fPlfms,fProjGDC)
      fPlfms <- as.data.frame(read.csv('data/platforms.csv',stringsAsFactors = F,header=T))
      if (!exists('fPlfms'))
      fPlfms <- defGPL

      fprojGDC <- as.data.frame(read.csv('data/projGDC.csv',stringsAsFactors = F,header=T))
      if (!exists('fprojGDC')){
        fprojGDC <- as.data.frame(TCGAbiolinks::getGDCprojects())[,c(6,4,5)]
        if(!exists('fProjGDC'))
          fprojGDC <- tGDC1
      }
    },silent=T)
  })


  #-----------------------------------------------------------------------------------

  # Listado de organismos involucrados (se toma GEO como referencia)

  tOrgan <- unique(as.data.frame(fGEO('select organism from gpl')))

  tOrganParse <- new('array')

  for (i in 1:dim(tOrgan)[1]){

    tOrgan[i,] <- str_replace_all(tOrgan[i,],';\t','\n')
    temp <- stri_split_boundaries(tOrgan[i,],type='sentence',simplify=T)
    temp <- str_replace(temp,'\n','')
    tOrganParse <- append(tOrganParse,temp)
  }

  tOrgan <- unique(tOrganParse)

#-------------------------------------------------------------------------------------
# OTROS PARÁMETROS CONFIGURABLES DESDE LA APLICACIÓN

  # En principio, no configurable desde la aplicación (muestra)

  defSmpSel <- 300

  # Tamaño máximo por defecto. Toma como referencia el máximo de todos los experimentos de GEO
  # MEJORA: tener en cuenta el resto de repositorios para calcular el máximo
  # Alternativa: usar un valor razonable máximo (5000 aprox.)

  defSmpls <- 5000
  # defSmpls <- max(defGSE$Muestras),5000)

  # Centrada en organismo o plataforma

  defOrPla <- FALSE

  # Organismo por defecto

  defOrgan <- 'Homo sapiens'

  # Repositorio por defecto

  defRpstr <- 'GEO'

  # Tipo por defecto

  defpType <- 'Microarray'

  # Operador por defecto
```

```
  defOp <- 'AND'

  # Búsqueda por fecha de submission

  defFecha <- F

  # Límite de estudios para comparativas

  defComp <- 3

  # Límite de muestras para comparar

  defMaxMu <- 5

  # Descarga de datos raw/procesados

  defRawDa <- F

#--------------------------------------------------------------------------------
# VARIABLES DE SESIÓN

  # gMinSp: mínimo de muestras por defecto en la pantalla de selección
  # gMaxSp: máximo de muestras por defecto en la pantalla de selección
  # gDefPf: plataforma por defecto, se carga a partir de un fichero
  # gDefGD: tabla con los proyectos GDC (Prioridad: 1.Fichero, 2.API, 3.Memoria)
  # gDefRp: repositorio por defecto
  # gDefTp: tipo de experimento por defecto (ArrayExpress)
  # gKeywd: keywords por defecto
  # gSchop: operador de bsqueda
  # gOrgan: organismo por defecto
  # gOrPla: indica si la consulta GEO se centra en organismo o por plataforma involucrada
(cuando está por plataforma, organismo=Homo sapiens)
  # gDefFe: Usar fecha de publicación o submission (TRUE)
  # gCompa: Límite de estudios a comparar
  # gMaxMu: Límite de muestras a comparar
  # gRawDa: Descarga de datos raw o procesados

  # isBeta: indica si los ratios de metilación son beta o M (logarítmicos)

  control <-
reactiveValues(gMinSp=1,gMaxSp=defSmpls,gDefGD=fprojGDC,gDefPf=fPlfms$ID[1],gDefRp=defRpstr,


gKeywd='',gDefTp=defpType,gSchop=defOp,gDefFe=defFecha,gOrgan=defOrgan,gOrPla=defOrPla,gCompa=d
efComp,
    gMaxMu=defMaxMu,gRawDa=defRawDa,isBeta=T)

  datos <-
reactiveValues(gGSE=NULL,GSEsel=NULL,GSMsel=NULL,infoGSEs=NULL,infoGSMs=NULL,gHist='',GDC=NULL,


mensajes=NULL,notificaciones=NULL,procesados=NULL,valores=NULL,methContainer=NULL,
                           anotaciones=NULL)

  datosDMR <- reactiveValues(DMRs=NULL)


  source('src/filters.R',local=T)
  source('src/panels.R',local=T)
  source('src/buttons.R',local=T)
  source('src/analysis.R',local=T)
  source('src/stats.R',local=T)
  source('src/tables.R',local=T)

})
```

```
#--------------------------------------------------------------------------------
# filters.R
#--------------------------------------------------------------------------------

msjeGDC <- '¡ Se requiere especificar al menos un proyecto GDC !'

# Categorías GEO relacionadas con estudios de metilación
```

```
catGEO <- c('Methylation profiling by SNP array','Methylation profiling by array',
            'Methylation profiling by genome tiling array','Methylation profiling by high
throughput sequencing')

# Categorías AE relacionadas con estudios de metilación

catAE <- c('Methylation profiling by array','Methylation profiling by high throughput
sequencing','Bisulfite-seq','MeDIP-seq')
cats <-
c('Methylation+profiling+by+array','Methylation+profiling+by+high+throughput+sequencing','Bisul
fite-seq','MeDIP-seq')

# Categorías GDC sobre estudios de metilación (experimental_strategy)

catGDCDC <- c('DNA Methylation','DNA methylation')
catGDCTy <- c('Methylation Beta Value','Methylation beta value','Bisulfite sequence
alignment','Methylation percentage')
catGDCES <- c('Methylation Array','Methylation array','Bisulfite-Seq')

#-------------------------------------------------------------------------------------

  # Función para marcar el flag de acue a las diferentes tipologías
  # Los datos de partida deben tener campo Repositorio, Tipo y Flag

  marcar <- function(datos,repos){

    if (repos=='GEO'){

      indicesA <- which(grepl(catGEO[1],datos$Tipo,ignore.case=T)|
grepl(catGEO[2],datos$Tipo,ignore.case=T)|grepl(catGEO[3],datos$Tipo,ignore.case=T))
      indicesS <- which(grepl(catGEO[4],datos$Tipo,ignore.case=T))
    }

    if (repos=='ArrayExpress'){

      indicesA <- which(grepl(catAE[1],datos$Tipo,ignore.case=T))
      indicesS <- which(grepl(catAE[2],datos$Tipo,ignore.case=T)|
grepl(catAE[3],datos$Tipo,ignore.case=T)|grepl(catAE[4],datos$Tipo,ignore.case=T))
    }

    if (repos=='GDC'){

      indicesA <- which(grepl(catGDCES[1],datos$Tipo,ignore.case=T))
      indicesS <- which(grepl(catGDCES[3],datos$Tipo,ignore.case=T))
    }

    indicesAS <- intersect(indicesA,indicesS)
    indicesAtot <- setdiff(indicesA,indicesAS)
    indicesStot <- setdiff(indicesS,indicesAS)

    if (length(indicesAtot)>0)
      datos[indicesAtot,8] <- 'A'
    if (length(indicesStot)>0)
      datos[indicesStot,8] <- 'S'
    if (length(indicesAS)>0)
      datos[indicesAS,8] <- 'AS'

    return(datos)
}


  # Función para extraer información de la respuesta a la query (XML)

  extractXML <- function(datos){

    xml <-xmlTreeParse(datos,useInternalNodes = T)
    # ID, nombre, fecha
    top <- xmlRoot(xml)
    xmlID <- getNodeSet(top,'experiment/accession')
    xmlNo <- getNodeSet(top,'experiment/name')
    xmlFe <- getNodeSet(top,'experiment/releasedate')
    ID <- unlist(as.vector(xmlApply(xmlID,xmlValue)))
    No <- unlist(as.vector(xmlApply(xmlNo,xmlValue)))
```

```
    Fe <- unlist(as.vector(xmlApply(xmlFe,xmlValue)))

    #Tipo
    long <- length(xmlID)
    Ti <- array(rep('',long))
    for (i in 1:long){

      vector <- as.vector(getNodeSet(top[[i]],'child::experimenttype'))
      df <- xmlToDataFrame(vector,stringsAsFactors = F)
      df <- as.array(df$text)
      str <- ''
      for (j in 1:length(df)){
        if (j<length(df))
          str <- paste(str,df[j],', ',sep='')
        else
          str <- paste(str,df[j],sep='')
      }
      Ti[i] <- str
    }

    #Plataforma: ¡¡ las plataformas de secuenciación no figuran directamente en la query XML!!

    xmlPl <- getNodeSet(top,'experiment/arraydesign/name')
    Pl <- array(rep('',long))

    if (!is.null(xmlPl)){

      for (i in 1:long){

        vector <- as.vector(getNodeSet(top[[i]],'child::arraydesign/name'))
        if (!is.null(vector)){
          df <- xmlToDataFrame(vector,stringsAsFactors = F)
          df <- parseIn('',df$text,'','',';','')
          Pl[i] <- df
        }
      }
    }
    else
      Pl[i] <- ''

    # Muestras (se toma como el máximo entre las dos tags, ya que si el experimento no es de
microarray no tiene el campo arraydesign)
    xmlSm <- getNodeSet(top,'experiment/bioassaydatagroup/bioassays')
    xmlSm2 <- getNodeSet(top,'experiment/arraydesign/counts')
    Sm <- array(rep(0,long))

    for (i in 1:long){
      Sm[i] <- 0
      temp <- 0
      if (!is.null(xmlSm)){
        vector <- as.vector(getNodeSet(top[[i]],'child::arraydesign/count'))
        if (!is.null(vector)){
          df <- xmlToDataFrame(vector,stringsAsFactors = F)
          df <- as.array(df$text)
          df <- as.integer(df)
          Sm[i] <- sum(df)
        }
      }

      if (!is.null(xmlSm2)){
        vector2 <- as.vector(getNodeSet(top[[i]],'child::bioassaydatagroup/bioassays'))
        if (!is.null(vector2)){
          df <- xmlToDataFrame(vector2,stringsAsFactors = F)
          df <- as.array(df$text)
          temp <- as.integer(df)
          Sm[i] <- max(Sm[i],temp)
        }
      }
    }

    try({

      final <- data.frame(ID,No,Fe,Ti,Pl,Sm,stringsAsFactors = F)
      final <- cbind(final,rep('ArrayExpress',dim(final)[1]))
```

```
      final <- cbind(final,rep('',dim(final)[1]))
      colnames(final) <-
c('ID','Nombre','Fecha','Tipo','Plataforma','Muestras','Repositorio','Flag')

    },silent=T)
    return(final)
  }

#--------------------------------------------------------------------------------

  # Filtra experimentos en GEO según los filtros de entrada. Devuelve dataset filtrado y
marcado

  filtroGEO <- function(keyWds,operador,iPlat,iSmps,iFech,iTipo,iOrga){

    subctoMet <- fGEO(paste("SELECT gse.gse AS GSE FROM gse
WHERE",parseLike('gse.type','Methylation profiling','')))

    if (control$gOrPla==T){
      if ((length(iPlat)>0)&&(!is.null(iPlat)))
        queryGEO <- fGEO(paste("SELECT gse.gse AS GSE,gse.title,gse.status,gse.type,gse_gpl.gpl
FROM gse JOIN gse_gpl ON gse.gse=gse_gpl.gse WHERE
gse_gpl.gpl",parseIn('IN',iPlat,'(',')',',',"'"),' ORDER BY gse.gse ASC'))
        queryGEO <- dplyr::inner_join(queryGEO,subctoMet,by='GSE')
    }

    else
      if ((length(iOrga)>0)&&(!is.null(iOrga))){

        # Equivalente a join de 3 tablas (más tiempo)
        queryTemp <- fGEO(paste("SELECT gse_gpl.gse,gse_gpl.gpl,gpl.title FROM gse_gpl,gpl
WHERE gse_gpl.gpl=gpl.gpl AND gpl.organism",parseIn('IN',iOrga,'(',')',',',"'"),' ORDER BY
gse_gpl.gse ASC'))
        if (dim(queryTemp)[1]>0){
          queryTemp <- unique(queryTemp[,2])
          queryGEO <- fGEO(paste("SELECT gse.gse as
GSE,gse.title,gse.status,gse.type,gse_gpl.gpl FROM gse JOIN gse_gpl ON gse.gse=gse_gpl.gse
WHERE gse_gpl.gpl",parseIn('IN',queryTemp,'(',')',',',"'"),'ORDER BY gse.gse ASC'))
          queryGEO <- dplyr::inner_join(queryGEO,subctoMet,by='GSE')
        }
      }

    if (dim(queryGEO)[1]>0){
      # Un mismo estudio puede usar distintas plataformas
      tempPlats <-
as.data.frame(geoConvert(queryGEO[,1],'gpl','data/GEOmetadb.sqlite'),stringAsFactors=F)
      colnames(tempPlats) <- c('GSE','GPL')

      tempTitles <- fGEO(paste("SELECT gpl.gpl,gpl.title FROM gpl WHERE
gpl",parseIn('IN',unique(tempPlats[,2]),'(',')',',',"'")))
      colnames(tempTitles) <- c('GPL','Nombre')

      tempPlats <- dplyr::inner_join(tempPlats,tempTitles,by='GPL')

      plats <- data.frame(count(tempPlats,GSE),'',stringsAsFactors=F)

      colnames(plats) <- c('GSE','Plats','Concat')

      for (i in 1:(dim(plats)[1])){
          indice <- which(tempPlats$GSE==plats$GSE[i])
          plats$Concat[i] <- parseIn('',tempPlats[indice,3],'','',';\t','')
      }

      plats <- plats[,-2]

      queryGEO <- dplyr::inner_join(queryGEO,plats,by='GSE')

      rm(tempPlats,tempTitles,plats)
      queryGEO[,5] <- queryGEO[,6]
      queryGEO <- queryGEO[,-6]
      queryGEO <- unique(queryGEO)

      querySam <- fGEO(paste("SELECT gse_gsm.gse,gse_gsm.gsm FROM gse_gsm WHERE
gse",parseIn('IN',queryGEO[,1],'(',')',',',"'")))
```

```
      Muestras <- table(sort(querySam[,1]))

      # Se añaden las variables Muestras, Repositorio y Flag (estadísticas y optimización)
      queryGEO <- cbind(queryGEO,as.array(Muestras))
      queryGEO <- cbind(queryGEO,rep('GEO',dim(queryGEO)[1]))
      queryGEO <- cbind(queryGEO,rep('',dim(queryGEO)[1]))
      queryGEO <- queryGEO[,-6]


      # Se 'formatea' la tabla
      colnames(queryGEO) <-
c('ID','Nombre','Fecha','Tipo','Plataforma','Muestras','Repositorio','Flag')
      queryGEO$Repositorio <-
factor(queryGEO$Repositorio,levels=c('GEO','ArrayExpress','TCGA'))
      queryGEO$Flag <- factor(queryGEO$Flag,levels=c('A','S','AS'))

      # Conversión de la variable Fecha (publicación). Si el parámetro fecha de registro está
activado se sustituye
      queryGEO$Fecha <-
paste(substr(queryGEO$Fecha,11,13),'/',substr(queryGEO$Fecha,15,16),'/',substr(queryGEO$Fecha,1
8,21),sep='')
      Sys.setlocale('LC_TIME','C')
      queryGEO$Fecha <- as.Date(queryGEO$Fecha,"%b/%d/%Y")
      Sys.setlocale('LC_TIME','')

      if (control$gDefFe){

        fechasSub <- fGEO(paste('SELECT gse.submission_date FROM gse WHERE gse.gse
',parseIn('IN',queryGEO$ID,'(',')',',',"'"),sep=''))
        queryGEO$Fecha <- fechasSub[,1]
      }

      # Marcado del campo Flag

      queryGEO <- marcar(queryGEO,'GEO')

      # FILTRADO: con los datos ya ajustados

      size <- length(keyWds)

      # Por tipo
      indiA <- which(queryGEO$Flag=='A')
      indiS <- which(queryGEO$Flag=='S')
      indiAS <- which(queryGEO$Flag=='AS')

      totalA <- union(indiA,indiAS)
      totalS <- union(indiS,indiAS)

      if (iTipo=='Microarray')
        queryGEO <- queryGEO[totalA,]
      if (iTipo=='Secuenciado')
        queryGEO <- queryGEO[totalS,]

      # Por muestras
      queryGEO <- queryGEO %>% filter(Muestras>=iSmps[1] & Muestras<=iSmps[2])

      # Por keywords
      if ((size>0)&&(keyWds!=''))
      {
        if (operador=='AND')
          for (i in 1:size)
          {
            queryGEO$match <- grepl(keyWds[i],queryGEO$Nombre,ignore.case = T)
            queryGEO <- queryGEO %>% filter(match==T)
            queryGEO <- queryGEO[,1:8]
          }
        if (operador=='OR')
        {
          for (i in 1:size)
          {
            match <- as.data.frame(grepl(keyWds[i],queryGEO$Nombre,ignore.case = T))
            colnames(match) <- paste('Match',i,sep='')
            queryGEO <- cbind(queryGEO,match)
          }
```

```
          if (size==1)
            queryGEO <- queryGEO %>% filter(queryGEO[,9])
          if (size==2)
            queryGEO <- queryGEO %>% filter(queryGEO[,9] | queryGEO[,10])
          if (size==3)
            queryGEO <- queryGEO %>% filter(queryGEO[,9] | queryGEO[,10] | queryGEO[,11])
          if (size==4)
            queryGEO <- queryGEO %>% filter(queryGEO[,9] | queryGEO[,10] | queryGEO[,11] |
queryGEO[,12])
          queryGEO <- queryGEO[,1:8]
        }
      }

      # Por fechas
      if (!is.na(iFech[1]))
        queryGEO <- queryGEO %>% filter(Fecha>=iFech[1])
      if (!is.na(iFech[2]))
        queryGEO <- queryGEO %>% filter(Fecha<=iFech[2])
    }

    return(queryGEO)
  }

#--------------------------------------------------------------------------------

  # Se encarga de obtener la información al repositorio AE, a partir de los filtros recogidos
en la interfaz de usuario

  filtroAE <- function(keyWds,operador,iSmps,iFech,iTipo,iOrga,iPlat){

    # Construcción de la query XML
    if (!is.null(keyWds)){

      size <- length(keyWds)

      query <- paste(urlAE)

      # Por keywords

      if (keyWds!=''){
        query <- paste(query,'keywords=',sep='')
        suf <- ''
        for (i in 1:length(keyWds)){
          if (i!=size)
            suf <- paste(suf,keyWds[i],'+',operador,'+',sep='')
          else
            suf <- paste(suf,keyWds[i],sep='')
        }
        query <- paste(query,suf,sep='')
      }
    }

    if (keyWds!='')
      query <- paste(query,'&',sep='')

    # Por tipo

    if (iTipo=='Todos')
      query <-
paste(query,'exptype="',catAE[1],'"+OR+"',catAE[2],'"+OR+"',catAE[3],'"+OR+"',catAE[4],'"',sep=
'')
    if (iTipo=='Microarray')
      query <- paste(query,'exptype="',catAE[1],'"',sep='')
    if (iTipo=='Secuenciado')
      query <- paste(query,'exptype="',catAE[2],'"+OR+"',catAE[3],'"+OR+"',catAE[4],'"',sep='')

    # Por muestras

    query <- paste(query,'&samplecount=[',iSmps[1],' TO ',iSmps[2],']',sep='')

    # Por fecha

    if (!is.na(iFech[1]))
      fecha1 <- as.character(iFech[1])
```

```
    else
      fecha1 <- '*'

    if (!is.na(iFech[2]))
      fecha2 <- as.character(iFech[2])
    else
      fecha2 <- '*'

    if ((fecha1!='*')||(fecha2!='*'))
      query <- paste(query,'&date=[',fecha1,' ',fecha2,']',sep='')

    if (control$gOrPla==T){

      accArray <- stri_replace_all(iPlat,'',fixed='GPL')
      accArray <- paste('A-GEOD-',accArray,sep='')
      query <- paste(query,'&array=',accArray,sep='')
    }
      # query <- paste(query,'&species=%22homo+sapiens%22')
    if (control$gOrPla==F)
      query <- paste(query,'&species=%22',str_replace_all(iOrga,' ','+'),'%22',sep='')

    # Obtención y parsing XML
    download.file(query,'data/AE.xml')

    dataXML <- NULL
    try({
      dataXML <- extractXML('data/AE.xml')

      if (file.exists('data/AE.xml'))
        file.remove('data/AE.xml')

      # ELIMINAR ENTRADAS gestionadas por GEO

      indices <- which(grepl('GEO',dataXML$ID))

      if (length(indices)>0)
        dataXML <- dataXML[-indices,]

      # Filtrado final, por número de muestras totales del experimento
      dataXML <- dataXML %>% filter(Muestras<=iSmps[2] & Muestras>=iSmps[1])

      dataXML <- marcar(dataXML,'ArrayExpress')

    },silent=T)

    return(dataXML)
  }

#-------------------------------------------------------------------------------------

  # Recupera información de GDC a partir de los filtros

  filtroGDC <- function(iSmps,iFech,iTipo,iClas,iLega){

    dataGDC <- NULL
    errorQuery <- T

    #-------------------------------------------------------------------------------------
      # algunos proyectos no tienen datos de arrays de metilación(ej: TARGET-AML)
      # experimental strategy determina el tipo de estudio ('Bisulfite-seq','Methylation
array')
      # MEJORA: implementar un mecanismo de espera para no saturar las búsquedas
      # El problema es que en la misma consulta no se pueden especificar legacy T/F
    #-------------------------------------------------------------------------------------

    # nl_ge_files = files() %>%
    #   filter(~   cases.samples.sample_type=='Solid Tissue Normal' &
    #            cases.project.project_id == 'TCGA-BRCA' &
    #            analysis.workflow_type == "HTSeq - Counts") %>%
    #   GenomicDataCommons::select(c(default_fields('files'),
    #                                'cases.case_id',
    #                                'cases.samples.sample_type')) %>%
    #   results_all() %>%
    #   as.data.frame()
```

```
    if (iClas=='TCGA (Todos)'){

      indices <- grep('TCGA',control$gDefGD$ID)
      iClas <- as.character(control$gDefGD[indices,'ID'])
    }


    # iClas <- c('TCGA-OV','TARGET-NBL')  ### para pruebas

    if (iLega)
      query <- paste("query <- GDCquery(project =
",parseIn('c',iClas,'(',')',',',"'"),",legacy=T,access='open',data.category='DNA
methylation')",sep='')
    else
      query <- paste("query <- GDCquery(project =
",parseIn('c',iClas,'(',')',',',"'"),",access='open',data.category='DNA Methylation')",sep='')

    try({
      eval(parse(keep.source=T,text=query))
      rdos <- getResults(query)
      errorQuery=F
    },silent=F)

    if (errorQuery){
      dataGDC <- NULL
      ventanaError('Error en la consulta GDC')
    }
    else{

      try({
        for (i in 1:length(iClas)){

          rdos1 <- rdos %>% filter(project==iClas[i])
          subArray <- rdos1 %>% filter(grepl(catGDCES[1],experimental_strategy,ignore.case =T))
          subSecue <- rdos1 %>% filter(grepl(catGDCES[3],experimental_strategy,ignore.case =T))

          ID <- iClas[i]
          # Nombre representativo del conjunto de muestras buscado
          cl <- control$gDefGD[which(control$gDefGD$ID==iClas[i]),]$Clase
          ps <- control$gDefGD[which(control$gDefGD$ID==iClas[i]),]$PS
          Nombre <- paste(cl,' - PS: ',ps,' ST: ',sep='')
          Repositorio <- 'GDC'
          Flag <- ''

          if (dim(subArray)[1]>0){

            pfA <- levels(factor(subArray$platform))
            esA <- levels(factor(subArray$experimental_strategy))
            tdA <- unique(as.character(subArray$tissue.definition))
            NombreA <- paste(Nombre,parseIn('',tdA,'','',',',''),sep='')
            # se toma la fecha de actualización (última muestra actualizada). Común a datos
legacy y no legacy
            FechaA <- max(as.character(as.Date(subArray$updated_datetime,format="%Y-%m-%d")))
            Fecha <- FechaA
            esA <- parseIn('',esA,'','',';','')
            TipoA <- esA
            pfA <- parseIn('',pfA,'','',';','')
            PlataformaA <- pfA
            MuestrasA <- dim(subArray)[1]
            Muestras <- MuestrasA
            temporalA <-
data.frame(ID,NombreA,Fecha,TipoA,PlataformaA,Muestras,Repositorio,Flag)
            colnames(temporalA) <-
c('ID','Nombre','Fecha','Tipo','Plataforma','Muestras','Repositorio','Flag')
            dataGDC <- rbind(dataGDC,temporalA)
          }

          if (dim(subSecue)[1]>0){

            pfS <- levels(factor(subSecue$platform))
            esS <- levels(factor(subSecue$experimental_strategy))
            tdS <- unique(as.character(subSecue$tissue.definition))
            NombreS <- paste(Nombre,parseIn('',tdS,'','',',',''),sep='')
```

```
            FechaS <- max(as.character(as.Date(subSecue$updated_datetime,format="%Y-%m-%d")))
            Fecha <- FechaS
            esS <- parseIn('',esS,'','',';','')
            TipoS <- esS
            pfS <- parseIn('',pfS,'','',';','')
            PlataformaS <- pfS
            MuestrasS <- dim(subSecue)[1]
            Muestras <- MuestrasS
            temporalS <-
data.frame(ID,NombreS,Fecha,TipoS,PlataformaS,Muestras,Repositorio,Flag)
            colnames(temporalS) <-
c('ID','Nombre','Fecha','Tipo','Plataforma','Muestras','Repositorio','Flag')
            dataGDC <- rbind(dataGDC,temporalS)
          }
        }

        # Marcado de flag
        if (!is.null(dataGDC))
          dataGDC <- marcar(dataGDC,'GDC')

        # Se filtra por fecha muestras y tipo iSmps,iFech,iTipo

        # Por muestras
        dataGDC <- dataGDC %>% filter(Muestras>=iSmps[1] & Muestras<=iSmps[2])

        # Por fechas
        if (!is.na(iFech[1]))
          dataGDC <- dataGDC %>% filter(Fecha>=iFech[1])
        if (!is.na(iFech[2]))
          dataGDC <- dataGDC %>% filter(Fecha<=iFech[2])

        # Por tipo

        indiA <- which(dataGDC$Flag=='A')
        indiS <- which(dataGDC$Flag=='S')
        indiAS <- which(dataGDC$Flag=='AS')

        totalA <- union(indiA,indiAS)
        totalS <- union(indiS,indiAS)

        if (iTipo=='Microarray')
          dataGDC <- dataGDC[totalA,]
        if (iTipo=='Secuenciado')
          dataGDC <- dataGDC[totalS,]

        # Almacenamiento para recopilación posterior de información
        temp <- data.frame(lapply(rdos, as.character), stringsAsFactors=FALSE)
        datos$GDC <- temp
        # write.csv(temp,file='data/GDC.csv')

        datos$GDC <- rdos

      },silent=F)
    }

    return(dataGDC)
  }
```

```
#--------------------------------------------------------------------------------
# panels.R
#--------------------------------------------------------------------------------

 # PANELES DINÁMICOS

  ayuda <- 'En diferentes colores se han marcado los diferentes elementos de la pantalla de
selección de muestras. En color rojo se han marcado los filtros en los que es necesario pulsar
los botones de búsqueda manual en el botón del sidebar o el que está situado debajo del
repositorio. Los elementos marcados en naranja reaccionan al realizar cambios y los marcados en
verde aplican sólo para filtrar muestras procedentes del repositorio GDC. Las keywords de
búsqueda tienen un máximo de 50 caracteres y 4 palabras y por defecto, tienen un criterio
restrictivo con el operador AND. Otros elementos a tener en cuenta durante el manejo de la
aplicación son los del sidebar, que indican en todo momento la cantidad de estudios y muestras
concretas que se han cargado, así como las notificaciones que pueden indicar posibles errores
```

```
en la carga de datos'


  txtHlp1 <- 'La fecha indicada indica el momento en el que se ha MODIFICADO alguna de las
muestras del proyecto'
  txtHlp2 <- 'Se muestran datos clasificados como OPEN'
  txtRaw <- 'Campo RAW: [1.RAW | 2.Procesados | 3.RAW/Procesados]'

  urlEstGEO <- 'https://www.ncbi.nlm.nih.gov/geo/summary/?type=series'
  urlEstAE <- 'https://www.ebi.ac.uk/arrayexpress/'
  urlEstGDC <- 'https://portal.gdc.cancer.gov/reports/data-download-statistics'

#-------------------------------------------------------------------------------

  output$sbEsts <- renderText({
    if (!is.null(datos$GSEsel))
      dim(datos$GSEsel)[1]
  })

  output$sbSmps <- renderText({
    if (!is.null(datos$GSMsel))
      dim(datos$GSMsel)[1]
  })


  # output$msjMenu <- renderMenu({
  #
  #   msjs <- function(){
  #
  #     if (!is.null(datos$mensajes))
  #       df <- apply(datos$mensajes, 1, function(row) {
  #         messageItem(from = row[[1]], message = row[[2]],time=row[[3]])
  #       })
  #   }
  #   dropdownMenu(type = 'messages', .list = msjs())
  # })

  output$notMenu <- renderMenu({

    nots <- function(){

      if (!is.null(datos$notificaciones))
        df <- apply(datos$notificaciones, 1, function(row) {
          notificationItem(text = row[[1]],status=row[[2]])
        })
    }
    dropdownMenu(type = 'notifications', .list = nots())
  })

  output$UIMain <- renderUI({

    fluidPage(
      conditionalPanel("input.menu=='opcion1'",uiOutput('UISel'),width='100%',height='auto'),
      conditionalPanel("input.menu=='opcion2'",uiOutput('UIDat'),width='100%',height='auto'),
      conditionalPanel("input.menu=='opcion3'",uiOutput('UIAna'),width='100%',height='auto'),
      conditionalPanel("input.menu=='opcion4'",uiOutput('UIVis'),width='100%',height='auto'),
      conditionalPanel("input.menu=='opcion5'",uiOutput('UICfg'),width='100%',height='auto'),
      conditionalPanel("input.menu=='opcion6'",uiOutput('UISta'),width='100%',height='auto'),
      conditionalPanel("input.menu=='opcion7'",uiOutput('UIHlp'),width='100%',height='auto')
    )
  })

#-------------------------------------------------------------------------------

  # output$notificaciones <-
renderDropdownMenu(notificationItem(text='ssf',icon=shiny::icon('cubes')))

#-------------------------------------------------------------------------------

  # 1.SELECCIÓN

  output$UISel <- renderUI({

    fluidRow(width='100%',
```

```
      column(12,tags$h4(align='center',renderText(m1msjI))),
      column(12,tags$br()),

column(2,selectInput('i1Reps','Repositorio',c('Todos','GDC','GEO','ArrayExpress'),selected =
control$gDefRp),
        actionButton('b1Lanz','Buscar',icon=shiny::icon('cloud'))),
        # actionButton('b1Canc','Cancelar',icon=shiny::icon('stop-circle'))),


column(2,conditionalPanel(condition=paste(str_to_lower(control$gOrPla),'==true',sep=''),selectI
nput('i1Pfms','Plataforma',fPlfms[,1],selected=control$gDefPf)),

conditionalPanel(condition=paste(str_to_lower(control$gOrPla),'==false',sep=''),selectInput('i1
Orga','Organismo',tOrgan,selected=control$gOrgan)),
        actionLink('l1Pfms','Detalles...')),

      column(2,{selectInput('i1Type','Tipo',c('Todos','Microarray','Secuenciado'),selected =
control$gDefTp)},
        actionLink('l1Type','Ayuda...')),

      column(3,dateRangeInput('i1Fech','Publicado en las
fechas',language='es',separator='a',min='2000-01-01',max=file.info('data/GEOmetadb.sqlite')
$ctime)),
      column(3,sliderInput('s1Smps','Muestras asociadas',min = control$gMinSp,max =
control$gMaxSp,step = 1,value=c(1,defSmpSel),dragRange=T)),
      column(12,conditionalPanel("input.i1Reps=='GDC' |
input.i1Reps=='Todos'",tags$br(),uiOutput('UISelGDC'))),
      column(12,dataTableOutput('d1Stud'),conditionalPanel("$
('#d1Stud').hasClass('recalculating')",tags$div('Loading ... '))),
      conditionalPanel("input.d1Stud_rows_selected.length > 0",
        column(6,actionButton('b1Selt','Seleccionar',icon=shiny::icon('crosshairs')),
          downloadButton('b1Desc','Exportar datos',icon=shiny::icon('download'))),

column(6,tags$div(id='logo2',align='right',actionButton('b1Limp','Limpiar',icon=shiny::icon('tr
ash')),
          actionButton('b1Todo','Todos',icon=shiny::icon('cubes')))))
      # column(3,offset=6,valueBoxOutput('v1Info',width = 12))
    )
  })

  # output$v1Info <- renderValueBox({
  #   if (!is.null(seleccionadosGSE()))
  #   valueBox(
  #     datos$gGSE[seleccionadosGSE(),'ID'],color =
'green',subtitle=paste(datos$gGSE[seleccionadosGSE(),'Muestras'],'muestras'),
  #     icon = icon("table"))
  # })


  output$UISelGDC <- renderUI({

    fluidRow(widt0='100%',


column(2,selectInput(width='100%','i1DisT','Clase',c(as.character(control$gDefGD$ID),'TCGA
(Todos)'),multiple=F,selected=control$ID[1]),
             actionLink('l1ClPS','Detalles...')),
      column(3,checkboxInput('c1Lega','Datos legacy (hg18,hg19)',value =
F),helpText(txtHlp1),tags$u(helpText(txtHlp2))),
      column(1,selectInput('i1TumT','Tipo
tumoral',as.array(tGDC2$ID),multiple=T),actionLink('l1SmTy','Tipos...')),
      column(1,selectInput('i1Gend','Sexo',catGendGDC,multiple=T)),
      column(2,selectInput('i1Race','Raza',catRaceGDC,multiple=T)),
      column(2,selectInput('i1Ethn','Etnia',catEthnGDC,multiple=T))
    )
  })


#--------------------------------------------------------------------------------
```

```
  # 2.DATOS (Selección de muestras)

  output$UIDat <- renderUI({

    fluidRow(width='100%',
      column(12,
        conditionalPanel(condition='input.d1Stud_rows_selected.length>0',
          box(title='Datos generales del estudio (1)',width=NULL,collapsible=T,collapsed=(!
length(seleccionadosGSE()<1)),status='success',
            column(7,tags$b('Nombre:'),datos$infoGSEs$Nombre[1]),
            column(3,tags$b('Tipo:'),datos$infoGSEs$Tipo[1]),

column(2,tags$b('Identificador:'),tags$a(datos$infoGSEs$ID[1],href=creaURL(datos$GSEsel[1,]),ta
rget="_blank")),tags$hr(),
            column(12,tags$b('Resumen:'),renderText({datos$infoGSEs$Resumen[1]}),tags$hr()),
            column(7,tags$b('Estrategia:'),datos$infoGSEs$Estrategia[1]),
            column(3,tags$b('Plataforma:'),isolate({datos$GSEsel$Plataforma[1]})),
            column(2,tags$b('Fecha:'),isolate({datos$GSEsel$Fecha[1]}))
          ),

          box(title='Seleccionar
muestras',width=NULL,collapsible=T,collapsed=T,status='success',

tags$div(id='tod1',align='right',actionButton('b2Lim1','Limpiar',icon=shiny::icon('trash')),
            actionButton('b2Tod1','Todas',icon=shiny::icon('database'))),
            dataTableOutput('d2Smp1')
          )
        ),

        conditionalPanel(condition='input.d1Stud_rows_selected.length>1',
          box(title='Datos generales del estudio (2)',width=NULL,collapsible=T,collapsed=(!
length(seleccionadosGSE()<2)),status='success',
            column(7,tags$b('Nombre:'),datos$infoGSEs$Nombre[2]),
            column(3,tags$b('Tipo:'),datos$infoGSEs$Tipo[2]),

column(2,tags$b('Identificador:'),tags$a(datos$infoGSEs$ID[2],href=creaURL(datos$GSEsel[2,]),ta
rget="_blank")),tags$hr(),
            column(12,tags$b('Resumen:'),renderText({datos$infoGSEs$Resumen[2]}),tags$hr()),
            column(7,tags$b('Estrategia:'),datos$infoGSEs$Estrategia[2]),
            column(3,tags$b('Plataforma:'),isolate({datos$GSEsel$Plataforma[2]})),
            column(2,tags$b('Fecha:'),isolate({datos$GSEsel$Fecha[2]}))
          ),

          box(title='Seleccionar
muestras',width=NULL,collapsible=T,collapsed=T,status='success',

tags$div(id='tod2',align='right',actionButton('b2Lim2','Limpiar',icon=shiny::icon('trash')),
            actionButton('b2Tod2','Todas',icon=shiny::icon('database'))),
            dataTableOutput('d2Smp2')
          )
        ),

        conditionalPanel(condition='input.d1Stud_rows_selected.length>2',
          box(title='Datos generales del estudio (3)',width=NULL,collapsible=T,collapsed=(!
length(seleccionadosGSE()<3)),status='success',
            column(7,tags$b('Nombre:'),datos$infoGSEs$Nombre[3]),
            column(3,tags$b('Tipo:'),datos$infoGSEs$Tipo[3]),

column(2,tags$b('Identificador:'),tags$a(datos$infoGSEs$ID[3],href=creaURL(datos$GSEsel[3,]),ta
rget="_blank")),tags$hr(),
            column(12,tags$b('Resumen:'),renderText({datos$infoGSEs$Resumen[3]}),tags$hr()),
            column(7,tags$b('Estrategia:'),datos$infoGSEs$Estrategia[3]),
            column(3,tags$b('Plataforma:'),isolate({datos$GSEsel$Plataforma[3]})),
            column(2,tags$b('Fecha:'),isolate({datos$GSEsel$Fecha[3]}))
          ),

          box(title='Seleccionar
muestras',width=NULL,collapsible=T,collapsed=T,status='success',

tags$div(id='tod3',align='right',actionButton('b2Lim3','Limpiar',icon=shiny::icon('trash')),
            actionButton('b2Tod3','Todas',icon=shiny::icon('database'))),
```

```
            dataTableOutput('d2Smp3')
          )
        ),

        conditionalPanel(condition='input.d1Stud_rows_selected.length>3',
          box(title='Datos generales del estudio (4)',width=NULL,collapsible=T,collapsed=(!
length(seleccionadosGSE()<1)),status='success',
            column(7,tags$b('Nombre:'),datos$infoGSEs$Nombre[4]),
            column(3,tags$b('Tipo:'),datos$infoGSEs$Tipo[4]),

column(2,tags$b('Identificador:'),tags$a(datos$infoGSEs$ID[4],href=creaURL(datos$GSEsel[4,]),ta
rget="_blank")),tags$hr(),
            column(12,tags$b('Resumen:'),renderText({datos$infoGSEs$Resumen[4]}),tags$hr()),
            column(7,tags$b('Estrategia:'),datos$infoGSEs$Estrategia[4]),
            column(3,tags$b('Plataforma:'),isolate({datos$GSEsel$Plataforma[4]})),
            column(2,tags$b('Fecha:'),isolate({datos$GSEsel$Fecha[4]}))
          ),

          box(title='Seleccionar
muestras',width=NULL,collapsible=T,collapsed=T,status='success',

tags$div(id='tod4',align='right',actionButton('b2Lim4','Limpiar',icon=shiny::icon('trash')),
            actionButton('b2Tod4','Todas',icon=shiny::icon('database'))),
            dataTableOutput('d2Smp4')
          )
        ),

        conditionalPanel(condition='input.d1Stud_rows_selected.length>4',
          box(title='Datos generales del estudio (5)',width=NULL,collapsible=T,collapsed=(!
length(seleccionadosGSE()<1)),status='success',
            column(7,tags$b('Nombre:'),datos$infoGSEs$Nombre[5]),
            column(3,tags$b('Tipo:'),datos$infoGSEs$Tipo[5]),

column(2,tags$b('Identificador:'),tags$a(datos$infoGSEs$ID[5],href=creaURL(datos$GSEsel[5,]),ta
rget="_blank")),tags$hr(),
            column(12,tags$b('Resumen:'),renderText({datos$infoGSEs$Resumen[5]}),tags$hr()),
            column(7,tags$b('Estrategia:'),datos$infoGSEs$Estrategia[5]),
            column(3,tags$b('Plataforma:'),isolate({datos$GSEsel$Plataforma[5]})),
            column(2,tags$b('Fecha:'),isolate({datos$GSEsel$Fecha[5]}))
          ),

          box(title='Seleccionar
muestras',width=NULL,collapsible=T,collapsed=T,status='success',

tags$div(id='tod5',align='right',actionButton('b2Lim5','Limpiar',icon=shiny::icon('trash')),
            actionButton('b2Tod5','Todas',icon=shiny::icon('database'))),
            dataTableOutput('d2Smp5')
          )
        )
      ),

column(12,conditionalPanel(condition='input.d1Stud_rows_selected.length>0',actionButton('b2Sele
','Confirmar',icon=shiny::icon('legal')))
      )

    )
  })


#-------------------------------------------------------------------------------

  # 3.ANÁLISIS

  output$UIAna <- renderUI({

    fluidRow(width='100%',

      conditionalPanel(condition='input.d1Stud_rows_selected.length>0',
        box(title='Muestras',width=NULL,collapsible=T,collapsed=(!
length(seleccionadosGSE()<1)),status='success',
          dataTableOutput('d3Mues'),helpText(txtRaw),
```

```
          actionButton('b3RawD','Descargar datos RAW',icon=shiny::icon('download'))),

        box(title='Resultados',width=NULL,status='success',collapsible=T,
          dataTableOutput('d3Proc'),
          tags$div(id='botD3',align='right',actionButton('b3Beta','Beta/M
values',icon=shiny::icon('exchange')),downloadButton('b3Desc','Exportar
datos',icon=shiny::icon('download'))),
          actionLink('l3Vals','Valores...'),
          helpText(ifelse(control$isBeta,'Beta values','M values')),
          column(2,selectInput('i3Anot','Campos:',colnames(datos$anotaciones[,-1]),multiple=T),
                 actionButton('b3Camp','Mostrar',icon=shiny::icon('eye'))),
          column(4,selectInput('i3gru1','Grupo 1',datos$GSMsel[,1],multiple = T),
                 actionButton('b3DMRA','DMRs',icon=shiny::icon('trello'))),
          column(4,selectInput('i3gru2','Grupo 2',datos$GSMsel[,1],multiple = T),
                 helpText('BumpHunter con 10 iteraciones y cutoff 0.2')),
          column(1,numericInput('t3linf','L.
inferior',min=0,max=1,value=0),actionButton('b3Filt','Filtrado',icon=shiny::icon('filter'))),
          column(1,numericInput('t3lsup','L.
superior',min=0,max=1,value=1),actionButton('b3Rest','Reset',icon=shiny::icon('refresh')))

        ),
        box(title='DMRs',width=NULL,status='success',collapsible=T,collapsed=F,
           dataTableOutput('d3DMRs'),downloadButton('b3Des2','Exportar
datos',icon=shiny::icon('download'))),

        box(title='Importar datos',width=NULL,status='success',collapsible=T,collapsed=T,
          column(4,
            fileInput('w3file', 'Importar datos:',
                      accept = c(
                        "text/csv",
                        "text/comma-separated-values,text/plain",
                        ".csv")),
            radioButtons('r3Sepa', 'Separador del fichero:', inline=T,
                               choiceNames = list('Tabulador',',',';','Espacio'),
                               choiceValues = list('\t',',',';',' ')
            ),
            checkboxInput('c3Head','Header',value = T),tags$br(),
            helpText('Usar "." como separador decimal'),
            actionButton('b3Impt','Importar',icon=shiny::icon('upload'))

          ),
          column(8,dataTableOutput('d3Impo'))
        )
      )
    )
  })

#-------------------------------------------------------------------------------------

  # 4.VISUALIZACIÓN

  output$UIVis <- renderUI({

    fluidRow(width='100%',

      conditionalPanel(condition='input.d1Stud_rows_selected.length>0',

        column(4,box(title='Densidades valores
beta',width=NULL,status='success',collapsible=T,collapsed=F,
               plotOutput('p4Gra1',width='100%'))),
        column(4,box(title='Valores
medios',width=NULL,status='success',collapsible=T,collapsed=F,
               plotOutput('p4Gra2'))),
        column(4,box(title='Valores
medios/grupo',width=NULL,status='success',collapsible=T,collapsed=F,
               plotOutput('p4Vis3'))),
        column(12,box(title='DMRs',width=NULL,status='success',collapsible=T,collapsed=F,
               plotOutput('p4Vis4')))
      )
    )
  })

#-------------------------------------------------------------------------------------
```

```
  # 5.CONFIGURACIÓN

  output$UICfg <- renderUI({

    fluidRow(width='100%',

      column(width=12,tags$b('OPCIONES:'),tags$br(),tags$br()),

      column(width=3,numericInput('t5iSmp','Muestras
(inf.):',control$gMinSp,min=1,max=control$gMaxSp,step=1),
        selectInput('i5pRpt','Repositorio por
defecto',c('GEO','ArrayExpress','GDC'),selected=control$gDefRp),
        selectInput('i5sOpe','Operador',c('AND','OR'),selected=control$gSchop),
        numericInput('t5Comp','Limitar estudios',control$gCompa,min=1,max=5,step=1),
        checkboxInput('c5pFSu','Fecha de registro (GEO)',control$gDefFe),
        checkboxInput('c5Cent','Centrada en plataforma',control$gOrPla),
        checkboxInput('c5RawD','Descargar datos raw',control$gRawData)),

      column(width=3,numericInput('t5sSmp',label=paste('Muestras (sup.): [ <=
',defSmpls,' ]',sep=''),control$gMaxSp,min=1,max=control$gMaxSp,step=1),
        selectInput('i5pTyp','Tipo por
defecto',c('Microarray','Secuenciado'),selected=control$gDefTp),
        selectInput('i5Orga','Organismo por defecto',tOrgan,selected=control$gOrgan),
        numericInput('t5Maxm','Limitar muestras',control$gMaxMu,min=1,max=10,step=1),
        conditionalPanel('input.c5Cent==true', selectInput('i5pPfm','Plataforma por defecto
(GEO)',fPlfms[,1],selected=control$gDefPf)),
        actionLink('l5Actu','Comprobar GEOmetadb')),

      column(12,tags$br()),
      column(width=12,actionButton('b5Mod1','Aplicar cambios',icon=shiny::icon('thumbs-o-up
')),
          actionButton('b5Mod2','Valores predeterminados',icon=shiny::icon('clipboard')))
    )
  })

#--------------------------------------------------------------------------------

  # 6.ESTADÍSTICAS

  output$UISta <- renderUI({

    fluidRow(width='100%',
      column(width=12,
        tabsetPanel('x6pTab',type='tabs',
          tabPanel('Generales',tags$br(),uiOutput('UIStat1')),
          tabPanel('Resultados',tags$br(),uiOutput('UIStat2')),
          tabPanel('GDC',tags$br(),uiOutput('UIStat3'))))
    )
  })

  output$UIStat1 <- renderUI({

    fluidRow(width='100%',
      column(4,box(width=12,title='Estudios
recopilados',verbatimTextOutput('t6Sta3'),collapsible=T,collapsed = F,solidHeader = TRUE,
status = 'success')),
      column(4,box(width=12,title = 'TOP 25
Topics',plotOutput('p6Sta1'),collapsible=T,collapsed=(is.null(datos$gGSE)),solidHeader = TRUE,
status = 'success')),
      column(4,box(width=12,collapsible=T,title='Historial (hasta 20
consultas)',verbatimTextOutput('t6Sta2'),solidHeader = TRUE, status = 'success')),
      column(12,box(width=12,collapsible=F,solidHeader = T, status = 'success',title='Enlaces a
los datos oficiales',
        tags$b(tags$a(href=urlEstGEO,target="_blank",'Gene Expression Omnibus')),tags$br(),
        tags$b(tags$a(href=urlEstAE,target="_blank",'ArrayExpress')),tags$br(),
        tags$b(tags$a(href=urlEstGDC,target="_blank",'The Cancer Genome Atlas'))))
    )
  })

  output$UIStat2 <- renderUI({

    fluidRow(width='100%',
      column(4,box(width=12,title = '# Muestras',plotOutput('p6Sta4'),collapsible=T,solidHeader
= TRUE, status = 'success'))
```

```
    )
  })

  output$UIStat3 <- renderUI({

    fluidRow(width='100%',
             column(4,box(width=12,title = 'Casos por
proyecto',plotOutput('p6Sta7'),collapsible=T,solidHeader = TRUE, status = 'success'),
              box(width=12,title = paste(input$i1DisT,'- Por
plataforma'),plotOutput('p6Sta5'),collapsible=T,collapsed=(is.null(datos$GDC)),solidHeader =
TRUE, status = 'success')),
             column(4,box(width=12,title = 'Casos por tipo de
dataset',plotOutput('p6Sta8'),collapsible=T,solidHeader = TRUE, status = 'success'),
              box(width=12,title = paste(input$i1DisT,'- Por tipo de
muestra'),plotOutput('p6Sta6'),collapsible=T,collapsed=(is.null(datos$GDC)),solidHeader = TRUE,
status = 'success'))
    )

  })

#-------------------------------------------------------------------------------------

  # 7.AYUDA

  # Explicar criterio para filtrar las keywords introducidas en la búsqueda

  # Explicar que el campo de búsqueda manual (texto) sólo funciona en la pantalla inicial

  output$UIHlp <- renderUI({

    fluidRow(width='100%',
      column(width=12,
        box(title='Ayuda',width=12,collapsible=T,solidHeader = T,status='success',tags$br(),
          tags$h4(actionLink('l7cPfm',paste('Consultar referencias de plataformas GEO.
Organismo seleccionado: ',input$i1Orga,sep=''))),tags$br(),
          tags$div(img(src ="ayuda.png")),tags$br(),tags$p(ayuda))
      )
    )
  })
```

```
#-------------------------------------------------------------------------------------
# analysis.R
#-------------------------------------------------------------------------------------

# GRÁFICAS DE METILACIÓN

#-------------------------------------------------------------------------------------

  # Densityplot de los datos procesados (s?lo betas)

  output$p4Gra1 <- renderPlot(res=100,{

    tabla <- datos$procesados
    isBeta <- control$isBeta

    if (!is.null(tabla)&&isBeta){

      columnas <- which(colnames(tabla)%in%datos$GSMsel[,1])
      nombreMuestras <- datos$GSMsel[,1]
      tabla <- tabla[,columnas]

      densityBeanPlot(dat = as.matrix(tabla),pal = terrain.colors(8))

    }
  })

#-------------------------------------------------------------------------------------
```

```
  # Valores medios de metilación

  output$p4Gra2 <- renderPlot(res=100,{

    dataframe <- datos$procesados
    isBeta <- control$isBeta

    if (!is.null(dataframe)){

      columnas <- which(colnames(dataframe)%in%datos$GSMsel[,1])
      nombreMuestras <- isolate({datos$GSMsel[,1]})
      dataframe <- dataframe[,columnas]

      if(isBeta)
        ggplot(stack(dataframe), aes(x = ind, y = values)) +
          geom_boxplot(fill=terrain.colors(length(dataframe))) + xlab('Muestras') + ylab('Beta
values') +
          theme(axis.text.x = element_text(angle = 45, hjust = 1))
      else
        ggplot(stack(dataframe), aes(x = ind, y = values)) +
          geom_boxplot(fill=terrain.colors(length(dataframe))) + xlab('Muestras') + ylab('M
values') +
          theme(axis.text.x = element_text(angle = 45, hjust = 1))

    }

  })

#-------------------------------------------------------------------------------

  # Valores medios por grupos (si se han definido)

  output$p4Vis3 <- renderPlot(res=100,{

    try({
      porgrupos <- datos$procesados
      gr1 <- input$i3gru1
      gr2 <- input$i3gru2
      isBeta <- control$isBeta

      inter <- intersect(gr1,gr2)

      if (!is.null(porgrupos)&&!is.null(gr1)&&!is.null(gr2)&&(length(inter)<1)){

        columnas <- which(colnames(porgrupos)%in%datos$GSMsel[,1])
        porgrupos <- porgrupos[,columnas]
        unico <- gather(porgrupos)
        indicesGr1 <- which(unico[,1] %in% gr1)
        indicesGr2 <- which(unico[,1] %in% gr2)
        unico[indicesGr1,1] <- 'Grupo 1'
        unico[indicesGr2,1] <- 'Grupo 2'

        if (isBeta){
          colnames(unico) <- c('Grupo','Beta')

          ggplot(unico, aes(x = Grupo, y = Beta)) +
            geom_boxplot(fill=terrain.colors(length(unico))) + xlab('Grupos') + ylab('M
values')
        }
        else{
          colnames(unico) <- c('Grupo','Mvalues')

          ggplot(unico, aes(x = Grupo, y = Mvalues)) +
            geom_boxplot(fill=terrain.colors(length(unico))) + xlab('Grupos') + ylab('M
```

```
values')
        }
      }
    },silent=T)
  })
```

```
#--------------------------------------------------------------------------------
# buttons.R
#--------------------------------------------------------------------------------


  # Ruta donde se almacenan las exportaciones de la aplicación
  pathExt <- 'ext'

  # Información del tipo de estudio

  msje1 <- 'Experimentos en los que se usan arrays, incluyendo SNP arrays'

  msje2 <- 'Experimentos en los que se hace uso de secuenciado, incluyendo Bisulfite-Seq y
MeDIP-Seq'

  protGDC <- "The DNA Methylation Liftover Pipeline uses data from the Illumina Infinium Human
Methylation 27 (HM27) and HumanMethylation450 (HM450) arrays to measure the level
    of methylation at known CpG sites as beta values, calculated from array intensities (Level
2 data) as Beta = M/(M+U). Using probe sequence information provided in the
    manufacturer's manifest, HM27 and HM450 probes were remapped to the GRCh38 reference genome
[1]. Beta values were inherited from existing TCGA Level 3 DNA methylation data
    (hg19-based) based on Probe IDs. Legacy data use GRCh18/19"

#--------------------------------------------------------------------------------

  # Construye URL para consultar más información del estudio seleccionado

  creaURL <- function(gse){

    url <- ''
    try({

      if (!is.null(gse)&&dim(gse)[1]>0){

        if (gse[,'Repositorio']=='GEO')
          url <- paste(urlConGEO,gse[,'ID'],sep='')
        if (gse[,'Repositorio']=='ArrayExpress')
          url <- paste(urlConAE,gse[,'ID'],sep='')
        if (gse[,'Repositorio']=='GDC')
          url <- paste(urlConGDC,gse[,'ID'],sep='')
      }
    },silent=T)

    return(as.character(url))
  }

#--------------------------------------------------------------------------------

  # Devuelve la tabla de muestras filtrada (pasada como argumento) en el repositorio GDC
  # Se podría modificar directamente, ya que se ha definido como un valor reactivo

  filtraMuestras <- function(dt,cl,bi)
  {
    if (!is.null(dt)){

      smTi <- input$i1TumT
      smGe <- input$i1Gend
      smRa <- input$i1Race
      smEt <- input$i1Ethn
      indices <- NULL

      # Filtrado por tipología, ya que se indica en la tabla (menos filas que mapear después)
      if (!is.null(smTi)){

        for (i in 1:length(smTi)){

          temp <- which(grepl(smTi[i],tGDC2[,1],ignore.case = T))
```

```
          indices <- as.array(c(indices,temp))
        }
      }

      # Por tipología
      if (!is.null(indices)){

        tipoS <- tGDC2[indices,2]
        totales <- NULL

        for (i in 1:length(indices)){

          indDT <- which(grepl(tipoS[i],dt$tissue.definition,ignore.case=T))
          totales <- c(totales,indDT)
        }

        dt <- dt[totales,]
      }

      # Mapeo para obtener información de la muestra y clínica (si la tiene)
      if (!is.null(cl)){

        bcrPatient <- as.character(stri_extract_all_regex(dt$cases,"TCGA-[:alnum:][:alnum:]-
[:alnum:][:alnum:][:alnum:][:alnum:]"))
        dt <- dt[,c('cases','platform','file_id','tissue.definition')]
        dt <- base::cbind(dt,bcrPatient,stringsAsFactors=F)

        clin <- cl[,c('bcr_patient_barcode','year_of_birth','gender','race','ethnicity')]
        dt <- dplyr::left_join(dt,clin,by=c('bcrPatient'='bcr_patient_barcode'))
      }

      # Util si se aplican filtros relacionados con características de la muestra
      # if (!is.null(bios))
      # {
      #   submitterid <- as.character(stri_extract_all_regex(dt$cases,"TCGA-[:alnum:][:alnum:]-
[:alnum:][:alnum:][:alnum:][:alnum:]-[:alnum:][:alnum:][:alnum:]"))
      #   dt <- cbind(dt,submitterid)
      #   dt <- left_join(dt,clin,by=c('submitterid'='submitter_id'))
      # }

      # Por género
      if (!is.null(smGe)){

        totales <- NULL

        for (i in 1:length(smGe)){

          indDT <- which(grepl(paste('^',smGe[i],sep=''),dt$gender,ignore.case=T))
          totales <- c(totales,indDT)
        }

        dt <- dt[totales,]
      }

      # Por raza
      if (!is.null(smRa)){

        totales <- NULL

        for (i in 1:length(smRa)){

          indDT <- which(grepl(smRa[i],dt$race,ignore.case=T))
          totales <- c(totales,indDT)
        }

        dt <- dt[totales,]
      }

      # Por etnia
      if (!is.null(smEt)){

        totales <- NULL

        for (i in 1:length(smEt)){
```

```
          indDT <- which(grepl(smEt[i],dt$ethnicity,ignore.case=T))
          totales <- c(totales,indDT)
        }

        dt <- dt[totales,]
      }
    }

    try({if (dim(dt)[1]==0) dt <- NULL},silent=T)
    return(dt)
  }


#-------------------------------------------------------------------------------

  # Consulta la información de los estudios seleccionados (descripción, muestras)

  obtInfo <- function(){

    datos$infoGSEs <- NULL
    datos$infoGSMs <- NULL
    datos$GSMsel <- NULL
    datos$importa <- NULL
    num <- seleccionadosGSE()
    msjeError <- 'Problemas al recopilar datos de'

    # Recopilación de información antes de entrar en el bucle
    if (num!=0&&length(num)>0)
      if(any(datos$GSEsel$Repositorio=='GDC')){
        projs <- datos$GSEsel[which(datos$GSEsel[,'Repositorio']=='GDC'),'ID']
        clin <- GDCquery_clinic(projs,"clinical")
        bios <- GDCquery_clinic(projs,"biospecimen")
      }

    for (i in 1:length(num)){

      query <- NULL
      query2 <- NULL

      if (datos$GSEsel$Repositorio[i]=='GEO'){
        tryCatch({

          query <- fGEO(paste("SELECT gse,title,type,summary,overall_design FROM gse WHERE
gse.gse='",datos$GSEsel[i,1],"'",sep=''))
          if (is.na(query$overall_design))
            query$overall_design <- 'Empty field'
          query2 <-  fGEO(paste("SELECT gsm.gsm AS ID,gsm.gpl AS Platform,gsm.title AS
Name,gsm.description AS Description,characteristics_ch1 AS Canal1,characteristics_ch2 AS Canal2
FROM gsm JOIN gse_gsm ON gsm.gsm=gse_gsm.gsm WHERE
gse_gsm.gse='",datos$GSEsel[i,1],"'",sep=''))

        },error=function(e){

          msjeError <- paste(msjeError,'GEO')
          ventanaError(msjeError)
        })
      }

      if (datos$GSEsel$Repositorio[i]=='ArrayExpress'){

        tryCatch({

          query <-
data.frame(datos$GSEsel[i,'ID'],datos$GSEsel[i,'Nombre'],datos$GSEsel[i,'Tipo'])
          urlIdf <-
paste('https://www.ebi.ac.uk/arrayexpress/files/',datos$GSEsel[i,1],'/',datos$GSEsel[i,1],'.idf
.txt',sep='')
          urlSdrf <-
paste('https://www.ebi.ac.uk/arrayexpress/files/',datos$GSEsel[i,1],'/',datos$GSEsel[i,1],'.sdr
f.txt',sep='')
          download.file(urlIdf,paste('data/idf_',i,'.txt',sep=''))
          download.file(urlSdrf,paste('data/sdrf_',i,'.txt',sep=''))
          info <- read.csv(paste('data/idf_',i,'.txt',sep=''),header=F,stringsAsFactors =
```

```
F,sep='\t')
          muestras <-
read.csv(paste('data/sdrf_',i,'.txt',sep=''),sep='\t',header=T,stringsAsFactors = F)

          # Campo resumen
          indice <- which(grepl('Experiment Description',info[,1]))
          query$Resumen <- info[indice,2]

          # Campo diseño experimental
          indices <- which(grepl('Experimental Design',info[,1]))
          indiceFinal <- indices[1]
          noVacios <- which(!is.na(info[indiceFinal,-1])&&(info[indiceFinal,-1]!=''))
          if (length(noVacios)==0)
            query <- cbind(query,'Empty field')
          else{
            campo <- parseIn('',info[indiceFinal,-1],'','','','')
            campo <- stri_replace_all(campo,fixed = ';',replacement='')
            campo <- stri_replace_all(campo,fixed = 'NA',replacement='')
            query <- cbind(query,campo)
          }

          # Se sustituye por el del archivo debido a que en la query no son accesibles
plataformas de secuenciación (XML)
          indice <- which(grepl('Protocol Hardware',info[,1]))

          if (length(indice)!=0)
            campo <- info[indice,2]
          else
            campo <- ''

          if (length(campo)!=0&&!is.na(campo)&&(campo!=''))
            datos$GSEsel$Plataforma[i] <- campo


          # MUESTRAS
          # --------

          # Si hay datos procesados están en el campo Derived.Array.Data.File y
Comment..Derived.ArrayExpress.FTP.file.
          # Los datos raw están en el campo Array.Data.File y Comment..ArrayExpress.FTP.file.
          indices <- which(grepl('Source.name',colnames(muestras),ignore.case=T))
          indices <- c(indices,which(grepl('Array.Design',colnames(muestras),ignore.case=T)))
          # indices <-
c(indices,which(grepl('Derived.Array.Data.File',colnames(muestras),ignore.case=T)))
          # indices <-
c(indices,which(grepl('Comment..ArrayExpress.FTP.file.',colnames(muestras),ignore.case=T)))

          # Los estudios de secuenciación tiene el campo nombre diferentea los de microarray
          if (datos$GSEsel$Flag[i] %in% c('S','AS'))
            indices <- c(indices,which(grepl('Assay.Name',colnames(muestras),ignore.case=T)))
          if (datos$GSEsel$Flag[i] %in% c('A','AS'))
            indices <-
c(indices,which(grepl('Hybridization.Name',colnames(muestras),ignore.case=T)))

          indices <- c(indices,which(grepl('Factor',colnames(muestras),ignore.case=T)))
          indices <- c(indices,which(grepl('Label',colnames(muestras),ignore.case = T)))

          query2 <- muestras[,indices]


          colnames(muestras) <- stri_replace_all(colnames(muestras),fixed='.',' ')

          # Borrado de archivos temporales (IDF, SDRF)
          try({

            nom <- paste('data/idf_',i,'.txt',sep='')
            nom2 <- paste('data/sdrf_',i,'.txt',sep='')
            if (file.exists(nom))
              file.remove(nom)
            if (file.exists(nom2))
              file.remove(nom2)

          },silent=T)
```

```
        },error=function(e){

          msjeError <- paste(msjeError,'ArrayExpress')
          ventanaError(msjeError)
        })
      }

      if (datos$GSEsel$Repositorio[i]=='GDC'){
        OK <- F
        try({
          OK <- TCGAbiolinks::isServeOK()
        },silent=T)

        if (OK){

          tryCatch({

            query <-
data.frame(datos$GSEsel[i,'ID'],datos$GSEsel[i,'Nombre'],datos$GSEsel[i,'Tipo'])
            porMuestras <- ''
            estrategia <- ''
            tabla <- datos$GDC

            if (!is.null(tabla)){

              tabla <- tabla %>% filter(experimental_strategy==datos$GSEsel[i,'Tipo'])
              tissues <- as.data.frame(table(tabla$tissue.definition))
              campoDes <- 'Los archivos disponibles de acceso no controlado de acuerdo a los
tipos tumorales son los siguientes:'
              frecs <- paste(tissues[,1],tissues[,2])
              frecs <- parseIn('',frecs,'','',';\t','')
              frecs <- paste(campoDes,frecs)
              estrategia <- datos$GSEsel[i,'Tipo']
            }
            query$Resumen <- frecs
            query$Estrategia <- estrategia

            # Filtros de entrada para acotar resultados
            tabla <- filtraMuestras(tabla,clin,bios)
            query2 <- tabla
            # query2 <-
tabla[,c('file_id','cases','tissue.definition','gender','race','ethnicity','year_of_birth')]

          },error=function(e){

            msjeError <- paste(msjeError,'GDC')
            ventanaError(msjeError)
          })
        }
        else{
          msjeError <- paste(msjeError,'GDC')
          ventanaError(msjeError)
        }
      }

      # Asignación final y acciones comunes
      if (!is.null(query)&&(dim(query)[2]==5)){

        colnames(query) <- c('ID','Nombre','Tipo','Resumen','Estrategia')
        datos$infoGSEs <- rbind(datos$infoGSEs,query)
      }

      try({if(dim(query2)[1]==0) query2 <- NULL},silent=T)

      if (!is.null(query2))
        datos$infoGSMs[[i]] <- query2

    }
  }

#-------------------------------------------------------------------------------

  # Comprueba que las muestras procedan de plataformas admitidas, a efectos de ser comparables
```

```
  checkPerm <- function(){

    # Se procesa el fichero de plataformas de forma que se separan las distintas plataformas

    isolate({dataframe <- datos$GSEsel})

    if (!is.null(fPlfms)){

      # Permitidas




almacena un array de todas las plataformas permitidas indicadas en el fichero
      permitidas <- as.array(fPlfms[which(fPlfms[,1]!=''),1])
      eqAE <- fPlfms[which(fPlfms[,3]!=''),3]
      AE <- NULL
      for (j in 1:length(eqAE)){
        split <- str_split(eqAE[j],',',simplify=T)
        AE <- c(AE,split)
      }

      permitidas <- c(permitidas,AE)
      eqGDC <- fPlfms[which(fPlfms[,4]!=''),4]
      permitidas <- c(permitidas,eqGDC)
      permitidas <- unique(permitidas)

    }

    for (i in 1:dim(dataframe)[1]){
      if (!is.null(input$d2Smp1_rows_selected))
        num1 <- length(input$d2Smp1_rows_selected)

      dataframe[i,'Correcto'] <- T
      comando <- paste('num <- input$d2Smp',i,'_rows_selected',sep='')
      eval(parse(keep.source=T,text=comando))
      if (!is.null(num)){

        repo <- dataframe[i,'Repositorio']
        isolate({muestras <- datos$infoGSMs[[i]]})
        muestrasSel <- muestras[num,]

        plats <- NULL
        # Se podría unificar, ya que la plataforma de cada muestra está definida en la segunda
columna.
        if (repo=='GEO'){

          # Se comprueba que las muestras corresponden a la misma plataforma (un mismo gse
puede albergar muestras de distintas plat.)
          query <- fGEO(paste('SELECT DISTINCT gpl FROM gsm WHERE
gsm',parseIn('IN',muestrasSel$ID,'(',')',',',"'")))
          plats <- c(plats,query)
        }

        if (repo=='ArrayExpress')
          plats <- c(plats,muestrasSel[,2])
        if (repo=='GDC'){
          plats <- c(plats,muestrasSel[,2])
        }
        plats <- unique(plats)

        # Se comparan las plataformas de las muestras seleccionadas con las de las permitidas
        if (plats %in% permitidas){
          dataframe[i,'Correcto'] <- T
        }
        else{
          dataframe[i,'Correcto'] <- F
        }
      }
    }
    return(dataframe$Correcto)
  }
```

```
#-------------------------------------------------------------------------------------

  # Comprueba que las plataformas son equivalentes. Devuelve T o F

  checkEquiv <- function(){

    # Equivalencias almacena cada equivalencia en un string "GPL8490,A-GEOD-8490,A-MEXP-2255"

    isolate({dataframe <- datos$GSEsel})
    plats <- NULL

    for (i in 1:dim(dataframe)[1]){

      comando <- paste('num <- input$d2Smp',i,'_rows_selected',sep='')
      eval(parse(keep.source=T,text=comando))
      if (!is.null(num)){

        repo <- dataframe[i,'Repositorio']
        isolate({muestras <- datos$infoGSMs[[i]]})
        muestrasSel <- muestras[num,]

        plats <- c(plats,muestrasSel[,2])

      }
    }
    plats <- unique(plats)

    correcto <- F
    if (!is.null(fPlfms)){

      equiv <- NULL
      for (j in 1:dim(fPlfms)[1]){
        equiv[j] <- parseIn('',c(fPlfms[j,1],fPlfms[j,3],fPlfms[j,4]),'','',',',',','')
        sep <- c(stri_split_fixed(equiv[j],','))
        if (all(plats %in% sep[[1]])==T)
          correcto <- T
      }
    }

    return(correcto)
  }

  #-------------------------------------------------------------------------------------

  # Descarga las muestras seleccionadas (previamente comprobadas)

  descargaMuestras <- function(){

    # Gracias a la caché de download.file no es necesario comprobar si se ha descargado
previamente el fichero (ej en único fichero RAW)
    dataframe <- isolate({datos$GSMsel})
    raw <- control$gRawDa # Posible uso futuro. Actualmente la opción de configuación no
permite cambiar a TRUE

    muestrasGEO <- filter(dataframe,Repositorio=='GEO')
    muestrasAE <- filter(dataframe,Repositorio=='ArrayExpress')
    muestrasGDC <- filter(dataframe,Repositorio=='GDC')

    if (dim(muestrasGEO)[1]>0){

      GSM <- muestrasGEO[,'ID']
      for (i in 1:length(GSM)){

        tryCatch({
          sup <- ''
          gsm <- getGEO(GEO=GSM[i],destdir=pathExt)
          sup <- parseIn('',gsm@header$supplementary_file,'','',',','')
          pro <- parseIn('',gsm@header$data_processing,'','','','')

          # Clasificación en función de los datos que posee
          if (dim(gsm@dataTable@table)[1]>0){
            if(sup=='NONE'){
              OKSup <- 2
```

```
            }
            else
              OKSup <- 3
          }
          else
            if(sup!='NONE')
              OKSup <- 1
            else
              OKSup <- 0

          # Se añade información adicional
          # Valores de OKSup: 1. Raw, 2. Procesados, 3. Raw/Procesados
          muestrasGEO$OKSup[i] <- OKSup
          muestrasGEO$sup[i] <- sup
          muestrasGEO$pro[i] <- pro

          # Comprobación del parámetro datos raw (DESHABILITADO TEMPORALMENTE EN OPCIONES)
          if (raw){ # RAW FALSE siempre hasta modificación futura

            if (OKSup %in% c(1,3)){
              gsmRaw <- getGEOSuppFiles(GSM[i],makeDirectory = F,baseDir = pathExt)
              gunzip(rownames(gsmRaw))
            }
            else
              datos$notificaciones <- rbind(datos$notificaciones,c(paste(GSM[i],'sin datos
RAW'),'warning'))
          }

          else{

            if (OKSup %in% c(2,3)){

              intensidades <- gsm@dataTable@table
              intensidades <- intensidades[,c(1,2)]
              colnames(intensidades) <- c('ID_REF',GSM[i])

              if (is.null(datos$procesados)){
                datos$procesados <- intensidades
                # colnames(datos$procesados)[-1] <- paste(GSM[i],'_',colnames(datos$procesados)
[-1],sep='')
              }

              else{

                # colnames(intensidades)[-1] <- paste(GSM[i],'_',colnames(intensidades)[-
1],sep='')
                # Al hacer el join podría haber sondas adicionales que se rellenan con NA
                datos$procesados <- dplyr::full_join(datos$procesados,intensidades,by='ID_REF')
              }

              descripcion <- parseIn('',gsm@dataTable@columns$Description,'','','',"'")
              datos$valores <- rbind(datos$valores,c(GSM[i],descripcion))
              colnames(datos$valores) <- c('ID','Valor')

            }
            else
              datos$notificaciones <- rbind(datos$notificaciones,c(paste(GSM[i],'sin datos
procesados'),'warning'))

          }

        },error=function(e){
          datos$notificaciones <- rbind(datos$notificaciones,c(paste('Error con
descarga',GSM[i]),'danger'))
        })
      }
    }

    # Muestras de ArrayExpress
    if (dim(muestrasAE)[1]>0){

      GSM <- muestrasAE[,'ID']
      # Algunos identificadores de muestra podrían interpretarse como enteros
      GSM <- as.character(GSM)
```

```
      for (i in 1:length(GSM)){

        tryCatch({

          sup <- ''
          OKSup <- 2

          urlSdrf <-
paste('https://www.ebi.ac.uk/arrayexpress/files/',muestrasAE[i,'IDGSE'],'/',muestrasAE[i,'IDGSE
'],'.sdrf.txt',sep='')
          urlIdf <-
paste('https://www.ebi.ac.uk/arrayexpress/files/',muestrasAE[i,'IDGSE'],'/',muestrasAE[i,'IDGSE
'],'.idf.txt',sep='')

          download.file(urlSdrf,paste(pathExt,'/sdrf.txt',sep=''),cacheOK = T)
          download.file(urlIdf,paste(pathExt,'/idf.txt',sep=''),cacheOK=T)

          muestras <-
read.csv(paste(pathExt,'/sdrf.txt',sep=''),sep='\t',header=T,stringsAsFactors = F)
          infoExp <-
read.csv(paste(pathExt,'/idf.txt',sep=''),sep='\n',header=T,stringsAsFactors = F)

          # Si las muestras son pareadas (doble canal), podrían existir dos filas con el mismo
source.name (único fichero)
          fila <- muestras[muestras$Source.Name==muestrasAE[i,'ID'],]

          # Estudios como E-MTAB-210 almacenan el nombre del archivo raw / procesado en
...Data.Matrix...
          indiceRaw <- which(grepl('^Array.Data.File',colnames(fila),ignore.case=T))
          indiceRaw2 <- which(grepl('^Array.Data.Matrix.File',colnames(fila),ignore.case=T))

          indiceProc <- which(grepl('^Derived.Array.Data.File',colnames(fila),ignore.case=T))
          indiceProc2 <-
which(grepl('^Derived.Array.Data.Matrix.File',colnames(fila),ignore.case=T))

          indiceFTPRaw <-
which(grepl('Comment..ArrayExpress.FTP.file.',colnames(fila),ignore.case=T))
          indiceFTPProc <-
which(grepl('Comment..Derived.ArrayExpress.FTP.file.',colnames(fila),ignore.case=T))

           if ((length(indiceRaw)>0)||(length(indiceRaw2)>0))
            if ((length(indiceProc)>0)||(length(indiceProc2)>0))
              OKSup <- 3
            else
              OKSup <- 1
          else
            if ((length(indiceProc)>0)||(length(indiceProc2)>0))
              OKSup <- 2
            else
              OKSup <- 0

          if (OKSup %in% c(1,3))
            if (length(indiceRaw)>0)
              sup <- unique(fila[,indiceRaw])
            else
              sup <- 'NONE'
          else
            sup <- 'NONE'

          filaPro <- which(grepl('Protocol description',infoExp[,1],ignore.case=T))
          infoExp <- infoExp[filaPro,1]

          # Debido a la extensión del campo se extraen los últimos 500 caracteres
          pro <- str_replace_all(infoExp,'Protocol Description','')
          if (nchar(pro)>500)
            pro <- substrRight(pro,500)
          pro <- paste('...',pro)

          muestrasAE$OKSup[i] <- OKSup
          muestrasAE$sup[i] <- sup
          muestrasAE$pro[i] <- pro

          # Comprobación del parámetro datos raw (DESHABILITADO TEMPORALMENTE EN OPCIONES)
```

```
          if (raw){
            if (OKSup %in% c(1,3)){
              datosExp <- getAE(muestrasAE[i,'IDGSE'],type='raw',path = pathExt)

            }

            else
              datos$notificaciones <- rbind(datos$notificaciones,c(paste(GSM[i],'sin datos
RAW'),'warning'))
          }

          else{

            if (OKSup %in% c(2,3)){

              # Con la función ArrayExpress no se descomprimen bien algunos archivos (ej: E-
MTAB-1529)
              # Se descargan del FTP directamente
              # datosExp <- getAE(dataframe[i,'IDGSE'],type='processed',path=pathExt,extract=F)

              # Dos posibilidades: único fichero con varias columnas (muestras) / un fichero
por muestra

              if (length(indiceProc)>0)
                columna <- 'Derived.Array.Data.File'
              else
                columna <- 'Derived.Array.Data.Matrix.File'

              # Puede haber estudios con tipo de datos procesado pero en formato .idat nativo
(ej: E-MTAB-5109)
              if (grepl('.idat',fila[,columna],ignore.case=T))
                datos$notificaciones <- rbind(datos$notificaciones,c(paste(GSM[i],'con formato
nativo'),'warning'))
              else{
                URLs <- unique(fila[,indiceFTPProc])
                nombreFich <- str_extract_all(URLs,pattern="E-[:alnum:][:alnum:][:alnum:]
[:alnum:]-[:alnum:][:alnum:][:alnum:][:alnum:].processed.[0-9].zip")

                if (!file.exists(paste(pathExt,'/',nombreFich,sep='')))
                  download.file(URLs,paste(pathExt,'/',nombreFich,sep=''),method='auto',cacheOK
= T)

                tryCatch({
                  ArrayExpress::extract.zip(paste(pathExt,'/',nombreFich,sep=''))

                },error=function(e){

                },finally={

                  unzip(paste(pathExt,'/',nombreFich,sep=''))
                })

                # Alternativa: read.columns de limma que permite seleccionar columnas
                # tryCatch({
                datosMet <-
limma::read.columns(paste(pathExt,'/',fila[,columna],sep=''),stringsAsFactors =
F,sep='\t',dec='.')
                # datosMet <-
limma::read.columns(paste(pathExt,'/',fila[,columna],sep=''),required.col=GSM[i],stringsAsFacto
rs = F,sep='\t')

                if (any(grepl('cg',rownames(datosMet))))
                  datosMet <- cbind(rownames(datosMet),datosMet)
                colnames(datosMet)[1] <- 'ID_REF'
                datosMet <- datosMet[,c('ID_REF',GSM[i])]
                if (dim(datosMet)[2]==2)
                  colnames(datosMet) <- c('ID_REF',GSM[i])

                # },error=function(e){
                #
                # },finally={
                #   datosMet <-
read.csv(paste(pathExt,'/',fila[,columna],sep=''),header=T,stringsAsFactors =
F,sep='\t',fileEncoding='UTF-16LE')
```

```
                # })

                # Selección de la columna de la muestra en el fichero de datos procesados
                # indiceMuestra <- which(grepl(GSM[i],colnames(datosMet)))
                # datosMet <- datosMet[,c(1,indiceMuestra)]
                intensidades <- datosMet

                if (is.null(datos$procesados)){

                  datos$procesados <- intensidades
                  colnames(datos$procesados)[-1] <- paste(GSM[i],'_',colnames(datos$procesados)
[-1],sep='')
                }

                else{

                  colnames(intensidades)[-1] <- paste(GSM[i],'_',colnames(intensidades)[-
1],sep='')
                  # Al hacer el join podría haber sondas adicionales que se rellenan con NA
                  datos$procesados <- full_join(datos$procesados,intensidades,by='ID_REF')
                }

                pro <- substrRight(pro,50)
                pro <- paste('...',pro,sep='')

                descripcion <- pro
                datos$valores <- rbind(datos$valores,c(GSM[i],descripcion))
                colnames(datos$valores) <- c('ID','Valor')

              }
            }

            else
              datos$notificaciones <- rbind(datos$notificaciones,c(paste(GSM[i],'sin datos
procesados'),'warning'))

          }

        },error=function(e){
          datos$notificaciones <- rbind(datos$notificaciones,c(paste('Error con
descarga',GSM[i]),'danger'))
        })
      }

    }

    if (dim(muestrasGDC)[1]>0){

      # Muestras de GDC
      GSM <- muestrasGDC[,3]

      # write.csv(datos$GDC,'tabla.csv')
      # Alternativa: descarga directa GET

      # urlPre <- 'https://api.gdc.cancer.gov/data/'
      # payload <- parseIn('',GSM,'','',',','')
      # URL <- paste(urlPre,payload,sep='')
      # download.file(URL,paste(pathExt,'',sep=''))

      # A diferencia de con los repositorios anteriores se descargan todos los ficheros de
forma simultánea
      GenomicDataCommons::gdcdata(GSM,destination_dir = pathExt,overwrite = T)
      ficheros <- dir(pathExt)
      coin <- NULL

      if (length(ficheros)>0){

        for (i in 1:dim(muestrasGDC)[1]){
          coin <- c(coin,grep(muestrasGDC[i,1],ficheros))
        }

        ficherosGDC <- ficheros[coin]

        muestrasGDC[,'OKSup'] <- 2
```

```
        muestrasGDC[,'sup'] <- 'NONE'
        muestrasGDC[,'pro'] <- protGDC
      }

      # El identificador de muestra está implícito en el nombre del fichero
      for (i in 1:length(ficherosGDC)){

        tryCatch({

          nombreMuestra <- str_extract_all(ficherosGDC[i],"TCGA.*.gdc")
          nombreMuestra <- str_replace_all(nombreMuestra,'.gdc','')
          tablaMet <- read.csv(paste(pathExt,'/',ficherosGDC[i],sep=''),stringsAsFactors =
F,sep='\t',dec='.')

          # Sólo se toma el nombre de las sondas y beta values (se anota al final, podría haber
datos de otros estudios/repositorios)
          tablaMet <- tablaMet[,1:2]

          colnames(tablaMet) <- c('ID_REF',nombreMuestra)

          if (is.null(datos$procesados))
            datos$procesados <- tablaMet

          else
            datos$procesados <- dplyr::full_join(datos$procesados,tablaMet,by='ID_REF')

          descripcion <- 'Calculated beta values mapped to genome, per sample'
          datos$valores <- rbind(datos$valores,c(nombreMuestra,descripcion))
          colnames(datos$valores) <- c('ID','Valor')

        },error=function(e){
          datos$notificaciones <- rbind(datos$notificaciones,c(paste('Error con
descarga',nombreMuestra),'danger'))
        })
      }
    }

    try({

      muestrasTotales <- NULL
      # Acciones comunes

      if (dim(muestrasGEO)[2]==10){
        colnames(muestrasGEO) <-
c('ID','Plataforma','X1','X2','Repositorio','Flag','GSE','Raw','URL','Datos procesados')
        muestrasTotales <- muestrasGEO
      }

      if (dim(muestrasAE)[2]==10){
        colnames(muestrasAE) <-
c('ID','Plataforma','X1','X2','Repositorio','Flag','GSE','Raw','URL','Datos procesados')
        muestrasTotales <- rbind(muestrasTotales,muestrasAE)
      }

      if (dim(muestrasGDC)[2]==10){
        colnames(muestrasGDC) <-
c('ID','Plataforma','X1','X2','Repositorio','Flag','GSE','Raw','URL','Datos procesados')
        muestrasTotales <- rbind(muestrasTotales,muestrasGDC)
      }

      datos$GSMsel <- muestrasTotales

      # Limpieza de archivos de la carpeta de descargas
      ficheros <- dir(pathExt)
      ind <- grep('RAW',ficheros)
      file.remove(paste(pathExt,'/',ficheros[-ind],sep=''))

    },silent=T)

    datos$notificaciones <- rbind(datos$notificaciones,c('Proceso de descarga
finalizado','success'))
  }

#-------------------------------------------------------------------------------
```

```
  # Validación básica de un fichero con datos procesados.
  # Requisitos: todos los campos numéricos y rownames o en primera columna contenga datos del
tipo cg...

  checkImpo <- function(df){

    # No se comprueba si existe porque ya se ha hecho desde la llamada (botón importar)

    valido <- F
    noNumericos <- NULL
    columnaID <- NULL

    # Identificación de campos no numéricos
    for (i in 1:length(colnames(df)))
      if (!is.numeric(df[,i]))
        noNumericos <- c(noNumericos,i)

    if (any(grepl('cg',rownames(df))))
      columnaID <- 0

    if (any(grepl('cg',df[,1])))
      columnaID <- 1

    if (is.null(noNumericos))
      if (!is.null(columnaID))
        if (columnaID==0)
          valido <- T

    if (!is.null(noNumericos)&&(!is.null(columnaID)))
      if (noNumericos==1&&columnaID==1)
        valido <- T

    return(valido)
  }

#-------------------------------------------------------------------------------------

  sanearProcesados <- function(){

    proc <- datos$procesados

    # La primera columna almacena el nombre de las sondas
    if (!is.null(proc)){

      nombreSondas <- proc[,1]
      nombreMuestras <- colnames(proc)
      proc <- as.data.frame(proc[,-1])
      minimos <- NULL

      # Si existen valores menores cero (al menos uno entre todas las sondas) se interpreta
como M values
      for (i in 1:dim(proc)[2]){
        vector <- proc[,i]
        minimos <- c(minimos,min(vector,na.rm=T))
      }

      columnas <- which(minimos<0)

      if (length(columnas)>0){

        betas <- function(x) (2^x)/((2^x)+1)
        # proc[,columnas] <- apply(proc[,columnas],1,function(x) (2^x)/((2^x)+1))
        proc[,columnas] <- unlist(lapply(proc[,columnas],betas))
      }

      # Se toman 7 cifras decimales (podría definirse un parámetro de configuración)
      # Motivo: se han observado beta values con distinto número de cifras significativas
(depende del preprocesamiento realizado)
      proc <- round(proc,digits=7)
      proc <- cbind(nombreSondas,proc)

      colnames(proc) <- nombreMuestras
      datos$procesados <- proc
```

```
      control$isBeta <- T
    }
  }

#---------------------------------------------------------------------------------

  # BOTONES Y LINKS


  # 0.SIDEBAR

  observeEvent(input$b0Selt,{

    if (input$menu=='opcion1'){

      control$gKeywd <- input$kw

      if (control$gKeywd!=''){

        if (length(datos$gHist)==20)
          datos$gHist <- datos$gHist[-1]

        datos$gHist <- append(datos$gHist,paste(control$gKeywd,'\n',sep=''))
      }
    }
  })

#---------------------------------------------------------------------------------

  # 1.SELECCIÓN

  # MEJORA: LOS BOTONES DE BORRAR LAS FECHAS NO ESTÁN IMPLEMENTADOS (OBSERVANDO), CUANDO SE
BORRA UNA FECHA NO MUESTRA RDOS.

  # Link plataformas GEO

  observeEvent(input$l1Pfms,{

    if ((length(input$i1Pfms)>0)&&(!is.null(input$i1Pfms)))
    {
      sql <- paste('SELECT gpl,title,technology,organism,manufacturer,bioc_package FROM gpl
WHERE gpl ',parseIn('IN',input$i1Pfms,'(',')',',',"'"),sep='')
      data <- fGEO(sql)
      colnames(data) <- c('ID','Nombre','Tipo','Organismo','Fabricante','Bioconductor')


showModal(modalDialog(renderDataTable(data,rownames=F,options=list(dom='tp'),selection='none'),
footer = NULL,
        title = 'Detalles de las plataformas',fade=TRUE,size='l',easyClose = T
      ))
    }
  })

  # Link tipos de estudios

  observeEvent(input$l1Type,{

    data <- data.frame(c('Microarray','Secuenciado'),c(msje1,msje2))
    colnames(data) <- c('Tipo','Almacena')


showModal(modalDialog(renderDataTable(data,rownames=F,options=list(dom='t'),selection='none'),f
ooter = NULL,
      title = 'Tipo de experimento',fade=TRUE,size='m',easyClose = T
    ))
  })

  # Link GDC proyectos

  observeEvent(input$l1ClPS,{

    data <- control$gDefGD


showModal(modalDialog(renderDataTable(data,rownames=F,options=list(dom='pt',pageLength=10),sele
```

```
ction='none'),footer = NULL,
      title = 'Equivalencia Clases - PS',fade=TRUE,size='m',easyClose = T
    ))
  })

  # Link GDC tipos de muestras

  observeEvent(input$l1SmTy,{

    data <- tGDC2


showModal(modalDialog(renderDataTable(data,rownames=F,options=list(dom='pt',pageLength=10),sele
ction='none'),footer = NULL,
      title = 'Tipos de muestras',fade=TRUE,size='s',easyClose = T
    ))
  })

  # Botón seleccionar estudio

  observeEvent(input$b1Selt,{

    num <- seleccionadosGSE()
    datos$GSEsel <- datos$gGSE[num,]

    if ((length(num)>0)&&(length(num)<=control$gCompa)){

      inhabConSel()
      try({obtInfo()},silent=T)
      datos$mensajes <- NULL
      datos$notificaciones <- NULL
      datos$GSMsel <- NULL
      datos$infoGSE <- NULL
      datos$infoGSM <- NULL
      actConSel()
      updateTabItems(session, 'menu','opcion2')
    }

    if (length(seleccionadosGSE()) >control$gCompa){

      msjeC <- paste('Se admiten hasta',control$gCompa,'estudios para realizar comparativas')
      ventanaError(msjeC)
    }

  })

  # Botón descargar tabla

  output$b1Desc <- downloadHandler(

    filename = function() { paste('GSEs_',Sys.Date(),'.csv',sep='') },

    content = function(file) {
      if (!is.null(datos$gGSE)){

        write.csv(datos$gGSE[,1:7],file,row.names=F)
      }
    }
  )

  # Botón limpiar selección

  observeEvent(input$b1Limp,{

    datos$GSEsel <- NULL
    datos$GSMsel <- NULL
    datos$procesados <- NULL
    datos$methContainer <- NULL
    datos$anotaciones <- NULL
    datosDMR$DMRs <- NULL
    datos$valores <- NULL
    datos$infoGSEs <- NULL
    datos$infoGSMs <- NULL
    reloadData(proxySel,clearSelection='all')
```

```
  })

  # Botón Seleccionar todos

  observeEvent(input$b1Todo,{

    selectRows(proxySel,todosGSE())

  })


  # Botón de búsqueda manual

  observeEvent(input$b1Lanz,{

    datos$gGSE <- NULL
    datos$GSEsel <- NULL
    datos$infoGSEs <- NULL
    datos$infoGSMs <- NULL
    datos$GSMsel <- NULL
    datos$valores <- NULL
    datos$procesados <- NULL
    datos$methContainer <- NULL
    datos$anotaciones <- NULL
    datosDMR$DMRs <- NULL
    control$gKeywd <- paste(control$gKeywd,' ',sep='')
  })

#-------------------------------------------------------------------------------------

  # 2. DATOS

  # Botones seleccionar todas las muestras

  observeEvent(input$b2Tod1,{

    selectRows(proxySmp1,todosSmp1())
  })

  observeEvent(input$b2Tod2,{

    selectRows(proxySmp2,todosSmp2())
  })

  observeEvent(input$b2Tod3,{

    selectRows(proxySmp3,todosSmp3())
  })

  observeEvent(input$b2Tod4,{

    selectRows(proxySmp4,todosSmp4())
  })

  observeEvent(input$b2Tod5,{

    selectRows(proxySmp5,todosSmp5())
  })


  # Botones de limpiar muestras

  observeEvent(input$b2Lim1,{

    # Poneer a NULL los objetos relacionados (seleccion de muestras GSMsel)

    reloadData(proxySmp1,clearSelection='all')
  })

  observeEvent(input$b2Lim2,{

    # Poneer a NULL los objetos relacionados (seleccion de muestras GSMsel)

    reloadData(proxySmp2,clearSelection='all')
```

```
})

observeEvent(input$b2Lim3,{

  # Poneer a NULL los objetos relacionados (seleccion de muestras GSMsel)

  reloadData(proxySmp3,clearSelection='all')
})

observeEvent(input$b2Lim4,{

  # Poneer a NULL los objetos relacionados (seleccion de muestras GSMsel)

  reloadData(proxySmp4,clearSelection='all')
})

observeEvent(input$b2Lim5,{

  # Poneer a NULL los objetos relacionados (seleccion de muestras GSMsel)

  reloadData(proxySmp5,clearSelection='all')
})

# Link para más información

observeEvent(input$l2Smp1,{

})

# Botón confirmar selección de muestras

observeEvent(input$b2Sele,{

  isolate({dataframe <- datos$GSEsel})
  datos$notificaciones <- NULL
  datos$GSMsel <- NULL
  datos$procesados <- NULL
  datos$methContainer <- NULL
  datos$anotaciones <- NULL
  datosDMR$DMRs <- NULL
  datos$valores <- NULL
  total <- 0
  num1 <- 0
  num2 <- 0
  num3 <- 0
  num4 <- 0
  num5 <- 0

  if (!is.null(input$d2Smp1_rows_selected))
    num1 <- length(input$d2Smp1_rows_selected)
  if (!is.null(input$d2Smp2_rows_selected))
    num2 <- length(input$d2Smp2_rows_selected)
  if (!is.null(input$d2Smp3_rows_selected))
    num3 <- length(input$d2Smp3_rows_selected)
  if (!is.null(input$d2Smp4_rows_selected))
    num4 <- length(input$d2Smp4_rows_selected)
  if (!is.null(input$d2Smp5_rows_selected))
    num5 <- length(input$d2Smp5_rows_selected)

  total <- num1+num2+num3+num4+num5

  if ((total>0)&&(total<=control$gMaxMu)){
  # Entra en el rango definido por el usuario

    inhabConSmp()
    correctos <- checkPerm()

    if (any(correctos)==F){
    # Alguno de los estudios tiene plataformas no incluidas en el fichero
```

```
        datos$notificaciones <- rbind(datos$notificaciones,c('Muestras con plataformas no
admitidas','warning'))
      }

      else{
      # Las muestras seleccionadas pertenecen a plataformas del fichero

        equivalentes <- checkEquiv()

        if (equivalentes){
        # Las muestras seleccionadas son equivalentes (pertenecen a una misma fila del fichero
de plataformas)

          # MUESTRAS DE ACUERDO A LOS CRITERIOS DE SELECCIÃ❽N
          for (i in 1:dim(dataframe)[1]){

            comando <- paste('num <- input$d2Smp',i,'_rows_selected',sep='')
            eval(parse(keep.source=T,text=comando))
            if (num!=0&&length(num)>0){

              isolate({muestras <- datos$infoGSMs[[i]]})
              muestrasSel <- muestras[num,1:4]
              muestrasSel[,1] <- as.character(muestrasSel[,1])
              muestrasSel[,2] <- as.character(muestrasSel[,2])
              muestrasSel[,3] <- as.character(muestrasSel[,3])
              muestrasSel[,4] <- as.character(muestrasSel[,4])
              muestrasRename <- cbind(muestrasSel,dataframe[i,'Repositorio'])
              muestrasRename <- cbind(muestrasRename,dataframe[i,'Flag'])
              muestrasRename <- cbind(muestrasRename,dataframe[i,'ID'])
              colnames(muestrasRename) <- c('ID','X1','X2','X3','Repositorio','Flag','IDGSE')
              datos$GSMsel <- dplyr::bind_rows(datos$GSMsel,muestrasRename)

            }
          }

          try ({

            descargaMuestras()

          # GSE47071 <- M VALUESrgSet <- read.450k.exp("GSE68777/idat")

            sanearProcesados()

          # Valores beta

            if (dim(nodescargados)[2]>0){
              nodescargados <- datos$GSMsel %>% filter(Raw==1)

showModal(modalDialog(renderDataTable(nodescargados,rownames=F,options=list(dom='tp'),selection
='none'),footer = NULL,
                                    title = 'Muestras sin datos
procesados',fade=TRUE,size='l',easyClose = T
              ))
              datos$GSMsel <- datos$GSMsel %>% filter(Raw==2 | Raw==3)
            }
            else{

            muestras <- datos$GSMsel
            betas <- datos$procesados
            betas <- as.data.frame(betas[,-1])
            betas <- as.matrix(betas)
            rownames(betas) <- datos$procesados[,1]

            logM <- function(x) log2(x/(1-x))
            mVals <- logM(betas)
            mVals <- round(mVals,digits=7)


            # Covariables
            # Se podrÃan desglosar las covariables, pero se ha diseÃ±ado por grupos
```

```
introducidos de forma manual
            covariables <- as.data.frame(datos$GSMsel[,3])
            colnames(covariables) <- 'Fenotipo'
            rownames(covariables) <- datos$GSMsel[,1]

            # OBJETO RATIOSET (sin anotaciones)
            temp <- RatioSet(Beta=betas,M = mVals,colData=covariables)


            # annotation(tmp) = annotation(MsetEx) (o directamente el .db)
            # tmp2 = mapToGenome(tmp)
            plats <- read.csv('data/platforms.csv')
            print(plats)
            platSmp <- muestras[1,2]
            fila <- which(plats[,1]==platSmp||plats[,3]==platSmp||plats[,4]==platSmp)

            idPlat <- plats[fila,6]
            idPlat <- str_replace(idPlat,'.db','')
            print(idPlat)
            rm(plats)
            annot <- NULL

            if (grepl('IlluminaHumanMethylation27k',idPlat,ignore.case=T)){
              try({
                annot <- read.table('data/HumanMethylation27_gdc_hg38.txt',sep='\t',header=T)
              },silent=T)
            }

            if (grepl('IlluminaHumanMethylation450k',idPlat,ignore.case = T)){
              try({
                annot <- read.table('data/HumanMethylation450_gdc_hg38.txt',sep='\t',header=T)
              },silent=T)
            }

            if (!is.null(annot)){

              colnames(annot)[1] <- 'ID'
              datos$anotaciones <- annot
              rm(annot)
            }
            datos$methContainer <- temp
            }
          },silent=T)
          updateTabItems(session, 'menu','opcion3')


        }
        else{
        # Las muestras seleccionadas no representan plataformas equivalentes (respecto al
fichero)

          datos$notificaciones <- rbind(datos$notificaciones,c('Plataformas no
equivalentes','warning'))
        }
      }

      # Se elimina la columna temporal
      last <- dim(dataframe)[2]
      dataframe <- dataframe[,-last]
      actConSmp()
    }

    else{
    # muestras seleccionadas fuera de los lÃmites

      if (total>0){

        msjeC <- paste('Se admiten hasta',isolate({control$gMaxMu}),'muestras en total de
cualquiera de los estudios seleccionados')
        ventanaError(msjeC)
      }
      else{
```

```
        msjeC <- paste('Se necesita al menos una muestra para analizar')
        ventanaError(msjeC)

      }
    }
  })

#-------------------------------------------------------------------------------

  # 3. ANÁLISIS

  # Link para mostrar qué representan los valores de intensidades de cada muestra

  observeEvent(input$l3Vals,{


showModal(modalDialog(renderDataTable(datos$valores,rownames=F,options=list(dom='tp'),selection
='none'),footer = NULL,
      title = 'Valores de intensidad de las muestras',fade=TRUE,size='l',easyClose = T
      ))

  })

  # Botón descargar tabla de intensidades

  output$b3Desc <- downloadHandler(

    filename = function() { paste('ValoresProcesados_',Sys.Date(),'.csv',sep='') },

    content = function(file) {
      if (!is.null(datos$procesados)){

        write.csv(datos$procesados,file,row.names=F)
      }
    }
  )

  # Botón descargar tabla DMRs

  output$b3Des2 <- downloadHandler(

    filename = function() { paste('DMRs_',Sys.Date(),'.csv',sep='') },

    content = function(file) {
      if (!is.null(datosDMR$DMRs)){

        write.csv(datosDMR$DMRs,file,row.names=F)
      }
    }
  )

  # Botón descargar datos RAW

  observeEvent(input$b3RawD,{

    shinyjs::disable('b3RawD')
    shinyjs::disable('b3Desc')
    shinyjs::disable('b3Impt')
    shinyjs::disable('w3file')

    dataframe <- isolate({datos$GSMsel})

    if (!is.null(dataframe)&&dim(dataframe)[2]==10){
      disponibleRAW <- filter(dataframe,(Raw==1 | Raw==3))
      contador <- dim(disponibleRAW)[1]
    }
    else
      contador <- 0

    if (contador>0){
      for (i in 1:contador){
```

```
        tryCatch({

          if (disponibleRAW[i,'Repositorio']=='GEO'){

            # OPCIÓN 1: Descarga "manual"
            # Extracción del nombre del fichero a partir de la URL (para descomprimir y
descargar)
            # URL <- disponibleRAW[i,'URL']
            # pos <- as.data.frame(str_locate_all(URL,'/'))
            # start <- pos[dim(pos)[1],2] + 1
            # nombreFich <- substr(URL,start,nchar(URL))
            #
            # download.file(URL,paste(pathExt,'/RAW/',nombreFich,sep=''))

            # OPCIÓN 2: Descarga asistida (getGEOSuppFiles)
            ficheros <- getGEOSuppFiles(disponibleRAW[i,'ID'],baseDir =
paste(pathExt,'/RAW/',sep=''),makeDirectory=F)

            for (j in 1:length(rownames(ficheros))){
              rownames(ficheros)[j] <- str_replace_all(rownames(ficheros)[j],'//','/')
              gunzip(rownames(ficheros)[j])
            }
          }

          if (disponibleRAW[i,'Repositorio']=='ArrayExpress'){

            ficheros <- getAE(disponibleRAW[i,'GSE'],type='raw',path =
paste(pathExt,'/RAW/',sep=''),extract = F)

            # Se elimina el archivo comprimido y el resto de ficheros descargados que no
contienen datos de metilación (.sdrf,.idf,.adf)
            # if (file.exists(ficheros$rawArchive))
            #   file.remove(ficheros$rawArchive)

            accesorios <- list.files(paste(pathExt,'/RAW/',sep=''))
            ind1 <- grep('.idf',accesorios)
            ind2 <- grep('.adf',accesorios)
            ind3 <- grep('.sdrf',accesorios)
            indices <- c(ind1,ind2,ind3)
            file.remove(paste(pathExt,'/RAW/',accesorios[indices],sep=''))
          }

          if (disponibleRAW[i,'Repositorio']=='GDC'){

            # Los datos RAW de GDC correspondientes a estudios de metilación son de tipo
legacy.

          }

        },error=function(e){
          datos$notificaciones <- rbind(datos$notificaciones,c(paste('Error con
descarga',disponibleRAW[i,'ID']),'danger'))
        })

      }
      datos$notificaciones <- rbind(datos$notificaciones,c('Proceso descarga RAW
finalizado','success'))
    }

    else
      datos$notificaciones <- rbind(datos$notificaciones,c('Muestras sin datos RAW
disponibles','warning'))

    shinyjs::enable('b3RawD')
    shinyjs::enable('b3Desc')
    shinyjs::enable('b3Impt')
    shinyjs::enable('w3file')

  })

  # Botón de importar datos procesados

  observeEvent(input$b3Impt,{
```

```
    shinyjs::disable('b3RawD')
    shinyjs::disable('b3Desc')
    shinyjs::disable('b3Impt')
    shinyjs::disable('w3file')

    dataframe <- isolate({importa()})
    columnas <- isolate({input$d3Impo_columns_selected})

    if (!is.null(dataframe)){

      correcto <- checkImpo(dataframe)

      if (correcto){

        if (any(grepl('cg',rownames(dataframe)))){
          dataframe <- c(rownames(dataframe),dataframe)
        }

        colnames(dataframe)[1] <- 'ID_REF'

        if (length(columnas>0)){
          # Dos posibilidades: datos cargados de alguna muestra o ningún dato cargado

          columnas <- c(1,columnas)

          # Datos previos
          if (!is.null(datos$procesados))
            datos$procesados <-
dplyr::full_join(datos$procesados,dataframe[,columnas],by='ID_REF')

          else
            datos$procesados <- dataframe[,columnas]

        }
        else
          ventanaError('No se han seleccionado columnas de muestras')
      }
      else
        ventanaError('Los datos cargados no son compatibles')
    }
    else
      ventanaError('No se han cargado datos')

    shinyjs::enable('b3RawD')
    shinyjs::enable('b3Desc')
    shinyjs::enable('b3Impt')
    shinyjs::enable('w3file')

  })

  # Botón de conversión beta/M values

  observeEvent(input$b3Beta,{

    formato <- isolate({control$isBeta})
    df <- isolate({datos$procesados})
    columnas <- which(colnames(df)%in%datos$GSMsel[,1])
    nombreColumnas <- colnames(df)
    numericos <- df[,columnas]

    if(formato){

      formula <- function(x) log2(x/(1-x))
      control$isBeta <- F
    }

    else{

      formula <- function(x) (2^x)/((2^x)+1)
      control$isBeta <- T
    }

    numericos <- formula(numericos)
```

```
  numericos <- round(numericos,digits=7)
  df[,columnas] <- numericos
  colnames(df) <- nombreColumnas
  datos$procesados <- df
})

# Botón para mostrar más información de las sondas (h38)

observeEvent(input$b3Camp,{

  campos <- isolate({input$i3Anot})
  anotaciones <- isolate({datos$anotaciones[,-1]})
  valores <- isolate({datos$procesados})

  if(!is.null(campos)&&!is.null(anotaciones)){

    columna1 <- which(colnames(valores)=='ID_REF')
    columnas2 <- which(colnames(valores)%in%datos$GSMsel[,1])

    subCto <- anotaciones[,campos]
    valores <- data.frame(valores[,columna1],subCto,valores[,columnas2])
    colnames(valores) <- c('ID_REF',campos,datos$GSMsel[,1])
  }

  else{

    columna1 <- which(colnames(valores)=='ID_REF')
    columnas2 <- which(colnames(valores)%in%datos$GSMsel[,1])
    valores <- data.frame(valores[,columna1],valores[,columnas2])
    colnames(valores) <- c('ID_REF',datos$GSMsel[,1])
  }

  datos$procesados <- valores

})

# Botón de filtrado

observeEvent(input$b3Filt,{

  inf <- isolate({input$t3linf})
  sup <- isolate({input$t3lsup})
  isB <- isolate({control$isBeta})
  dat <- isolate({datos$procesados})

  if (!isB)
    ventanaError('Convertir a beta values primero')
  else
    if ((inf>=0)&&(inf<=1)&&(sup>=0)&&(sup<=1)&&(inf<sup)){
      cols <- which(colnames(dat)%in%datos$GSMsel[,1])

      # Las columnas están en orden secuencial (3..5,2..8,4..5)
      for (i in min(cols):max(cols)){
        filas <- which(dat[,i]<=sup & dat[,i]>=inf)
        dat[-filas,i] <- NA
      }

    datos$procesados <- dat

    }

    else
      ventanaError('Los valores deben estar en el rango [0,1] y inf<sup')

})

# Botón de restablecer valores

observeEvent(input$b3Rest,{

  isB <- isolate({control$isBeta})
  dat <- isolate({datos$procesados})
  ori <- isolate({datos$methContainer})
  dat2 <- getBeta(ori)
```

```
    if (!isB)
      ventanaError('Convertir a beta values primero')
    else{

      cols1 <- which(colnames(dat)%in%datos$GSMsel[,1])
      cols2 <- which(colnames(dat2)%in%datos$GSMsel[,1])

      dat[,cols1] <- dat2[,cols2]
      datos$procesados <- dat

    }

  })

  # Botón de DMR

  observeEvent(input$b3DMRA,{

    isBeta <- isolate({control$isBeta})
    gr1 <- isolate({input$i3gru1})
    gr2 <- isolate({input$i3gru2})
    meth <- isolate({datos$methContainer})
    df <- isolate({datos$anotaciones})

    if (is.null(gr1)||(is.null(gr2)))
      ventanaError('Deben definirse los dos grupos a comparar')
    else{
      inters <- intersect(gr1,gr2)
      if (length(inters)>0)
        ventanaError('Los dos grupos deben ser excluyentes')
      else{

        inhabConAna()
        nombreMuestras <- union(gr1,gr2)
        feno <- data.frame(c(rep('grupo1',length(gr1)),rep('grupo2',length(gr2))))
        rownames(feno) <- nombreMuestras
        colnames(feno) <- 'fenotipo'
        design <- model.matrix(~feno$fenotipo)
        datos <- data.frame(chr=df$Chromosome,start=df$Start,end=df$End)
        rangos <- makeGRangesFromDataFrame(df, ignore.strand=TRUE)
        betas <- getBeta(meth)
        nuevo <- GenomicRatioSet(Beta=getBeta(meth),gr = rangos)
        if(isBeta)
          dmrs <- bumphunter(nuevo,design=design,cutoff = 0.2, B=10, type="Beta")
        else
          dmrs <- bumphunter(nuevo,design=design,cutoff = 0.2, B=10, type="M")
        datosDMR$DMRs <- dmrs$table
        actConAna()
      }
    }
  })

#-------------------------------------------------------------------------------------

  # 5.CONFIGURACIÓN

  # Función para aplicar los valores por defecto a los filtros

  valoresDef <- function(){

    control$gMinSp <- 1
    control$gMaxSp <- defSmpls
    control$gDefPf <- fPlfms
    control$gDefRp <- defRpstr
    control$gDefTp <- defpType
    control$gSchop <- defOp
    control$gDefFe <- defFecha
    control$gOrPla <- defOrPla
    control$gOrgan <- defOrgan
    control$gCompa <- defComp
    control$gMaxMu <- defMaxMu
    controldgRawDa <- defRawDa
  }
```

```
  # Valores de retorno: 0. Error al procesar variables de entrada, +1. Mínimo incorrecto, +10.
Máximo muestras, +100. Máximo estudios, 4. OK

  validPars <- function()
  {

    valid <- 0
    maxEst <- 5
    maxSmp <- 20

    # Apdo. Selecci?n

    try({

      i <- as.integer(isolate({input$t5iSmp}))
      s <- as.integer(isolate({input$t5sSmp}))
      c <- as.integer(isolate({input$t5Comp}))
      d <- as.integer(isolate({input$t5Maxm}))

      if ((i>=1)&&(i<s)&&(s<=defSmpls)&&(c>1)&&(c<=maxEst)&&(d>1)&&(d<=maxSmp)){

        control$gMinSp <- i
        control$gMaxSp <- s
        control$gDefPf <- isolate({input$i5pPfm})
        control$gDefRp <- isolate({input$i5pRpt})
        control$gDefTp <- isolate({input$i5pTyp})
        control$gSchop <- isolate({input$i5sOpe})
        control$gDefFe <- isolate({input$c5pFSu})
        control$gDefDe <- isolate({input$c5pDes})
        control$gOrPla <- isolate({input$c5Cent})
        control$gOrgan <- isolate({input$i5Orga})
        control$gCompa <- c
        control$gMaxMu <- d
        # control$gRawDa <- isolate({input$c5RawD}) # DESHABILITADO TEMPORALMENTE (siempre
FALSE)
        controldgRawDa <- defRawDa
        valid <- 4
      }

      else{
        if ((i<1)||(i>=control$gMaxSp))
          valid <- valid+1
        if (s>defSmpls)
          valid <- valid+10
        if ((c<=1)||(c>5))
          valid <- valid+100
        if ((d<=1)||(d>10))
          valid <- valid +1000
      }

    },silent=T)
    return(valid)
  }

  observeEvent(input$l5Actu,{

    pref <- 'La fecha de GEOmetadb es: '
    data <- paste(pref,file.info('data/GEOmetadb.sqlite')$mtime)

    showModal(modalDialog(renderText({data}),footer = NULL,
              title = 'GEOmetadb checking',fade=TRUE,size='s',easyClose = T
    ))
  })

  observeEvent(input$b5Mod1,{

    codigoError <- validPars()
    errorVal <- F

    if (codigoError==0){
      ventanaError('Error al validar')
    }
```

```
    if (codigoError==4){
      ventanaError('Se han modificado los valores')
      updateTabItems(session, 'menu','opcion1')
    }



    if (codigoError %in% c(1,11,101,111,1001,1011,1101,111)){
      errorVal <- T
      updateTextAreaInput(session,'t5iSmp',value = control$gMinSp)
    }
    if (codigoError %in% c(10,11,110,111,1010,1011,1110,1111)){
      errorVal <- T
      updateTextAreaInput(session,'t5sSmp',value = control$gMaxSp)
    }
    if (codigoError %in% c(100,101,110,111,1100,1101,1110,1111)){
      errorVal <- T
      updateTextAreaInput(session,'t5Comp',value = control$gCompa)
    }
    if (codigoError %in% c(1000,1001,1010,1011,1100,1101,1110,1111)){
      errorVal <- T
      updateTextAreaInput(session,'t5Maxm',value = control$gMaxMu)
    }

    if (errorVal)
      ventanaError('Valores incorrectos')

  })

  observeEvent(input$b5Mod2,{

    valoresDef()
    ventanaError('Se han aplicado los valores por defecto')
    # stopApp(returnValue = invisible())
    updateTabItems(session, 'menu','opcion1')


  })

#-------------------------------------------------------------------------------------

  # 7.AYUDA

  observeEvent(input$l7cPfm,{

    sql <- paste("SELECT gpl,title,technology,organism,manufacturer FROM gpl WHERE gpl.organism
LIKE '%",input$i1Orga,"%'",sep='')
    data <- fGEO(sql)
    colnames(data) <- c('ID','Nombre','Tipo','Organismo','Fabricante')


showModal(modalDialog(renderDataTable(data,rownames=F,options=list(autoWidth=T,scrollX=T,dom='f
tpi',pageLength=5),selection='none'),footer = NULL,
      title = 'Consulta de plataformas',fade=TRUE,size='m',easyClose = T
    ))
  })
```

```
#-------------------------------------------------------------------------------------
# stats.R
#-------------------------------------------------------------------------------------

# ESTADÍSTICAS

  #Crea una gráfica wordCloud

  plotWordCloud <- function(cadenas){

    corp <- VCorpus(VectorSource(cadenas))
    corp <- tm_map(corp,removePunctuation)
    corp <- tm_map(corp,content_transformer(tolower))
    corp <- tm_map(corp,removeNumbers)
    corp <- tm_map(corp,function(x)removeWords(x,stopwords()))
```

```
    term.matrix <- TermDocumentMatrix(corp)
    term.matrix <- as.matrix(term.matrix)
    v <- sort(rowSums(term.matrix),decreasing=T)
    d <- data.frame(word=names(v),freq=v)
    n <- 25
    pal <- brewer.pal(8,'Dark2')
    pal <- pal[-(1:4)]
    try({wordcloud(d[1:n,]$word,d[1:n,]
$freq,c(5,.4),min.freq=1,random.color=T,random.order=T,colors=pal)},silent=T)

  }

#-------------------------------------------------------------------------------------

  output$p6Sta1 <- renderPlot(res=100,{

    try({

    df <- datos$gGSE

    if(!is.null(df)){

      if (dim(df)[1]>=5)
      {
        words <- datos$gGSE$Nombre
        plotWordCloud(words)
      }
    }},silent=T)
  })

  output$t6Sta2 <- renderText({
    datos$gHist
  })

  output$t6Sta3 <- renderText({

    df <- datos$gGSE
    if(!is.null(df)&&dim(df)[1]>0){

      dfG <- df %>% filter(Repositorio=='GEO')
      dfA <- df %>% filter(Repositorio=='ArrayExpress')
      dfT <- df %>% filter(Repositorio=='GDC')
      dfGA <- dfG %>% filter(Flag=='A')
      dfGS <- dfG %>% filter(Flag=='S')
      dfGAS <- dfG %>% filter(Flag=='AS')
      dfAA <- dfA %>% filter(Flag=='A')
      dfAS <- dfA %>% filter(Flag=='S')
      dfAAS <- dfA %>% filter(Flag=='AS')
      dfTA <- dfT %>% filter(Flag=='A')
      dfTS <- dfT %>% filter(Flag=='S')
      dfTAS <- dfT %>% filter(Flag=='As')

      salida <- ''
      salida <- paste(salida,'# GEO: ',dim(dfG)[1],'\n',sep='')
      salida <- paste(salida,'\tMicroarray\t',dim(dfGA)[1],'\n',sep='')
      salida <- paste(salida,'\tSecuenciado\t',dim(dfGS)[1],'\n',sep='')
      salida <- paste(salida,'\tAmbos\t\t',dim(dfGAS)[1],'\n\n',sep='')
      salida <- paste(salida,'# ArrayExpress: ',dim(dfA)[1],'\n',sep='')
      salida <- paste(salida,'\tMicroarray\t',dim(dfAA)[1],'\n',sep='')
      salida <- paste(salida,'\tSecuenciado\t',dim(dfAS)[1],'\n',sep='')
      salida <- paste(salida,'\tAmbos\t\t',dim(dfAAS)[1],'\n\n',sep='')
      salida <- paste(salida,'# GDC: ',dim(dfT)[1],'\n',sep='')
      salida <- paste(salida,'\tMicroarray\t',dim(dfTA)[1],'\n',sep='')
      salida <- paste(salida,'\tSecuenciado\t',dim(dfTS)[1],'\n',sep='')
      salida <- paste(salida,'\tAmbos\t\t',dim(dfTAS)[1],'\n\n',sep='')
      salida <- paste(salida,'# Totales: ',dim(df)[1],'\n',sep='')
      salida <- paste(salida,'\tMicroarray\t',dim(dfTA)[1]+dim(dfGA)[1]+dim(dfAA)
[1],'\n',sep='')
      salida <- paste(salida,'\tSecuenciado\t',dim(dfTS)[1]+dim(dfGS)[1]+dim(dfAS)
[1],'\n',sep='')
      salida <- paste(salida,'\tAmbos\t\t',dim(dfTAS)[1]+dim(dfGAS)[1]+dim(dfAAS)
[1],'\n',sep='')
      salida
    }
```

```
  })

  output$p6Sta4 <- renderPlot(res=100,{

    df <- datos$gGSE

    if(!is.null(df)&&dim(df)[1]>0){

      ggplot(df,aes(Repositorio,Muestras)) + geom_boxplot(aes(fill=Repositorio))


    # Estadísticas de estudio con más y menos muestras (de los seleccionados), por repositorio
y
    # tipo

    # with(df,boxplot(Muestras~Repositorio))
    # ggplot(df,aes(Repositorio,Muestras)) + geom_boxplot(aes(fill=Repositorio))
    # ggplot(df,aes(Repositorio,Muestras)) + geom_boxplot()  + geom_jitter()
    # ggplot(df[1:25,],aes(Fecha,Muestras)) + geom_point(aes(color=Repositorio))

    # Enlace a las páginas con las estadísticas de los estudios almacenados (en cada
repositorio)


    # Muestras seleccionadas
    }
  })

  output$p6Sta5 <- renderPlot(res=100,{

    df <- datos$GDC

    if (!is.null(df)){

      frecs <- as.array(table(df$platform))
      frecs <- frecs[frecs>0]
      total <- sum(frecs)
      etiq <- names(frecs)
      etiq <- stri_replace_all(etiq,replacement = 'Illum. Human Meth',fixed='Illumina Human
Methylation')
      perc <- as.array(round((frecs/total)*100,digits = 2))
      labs <- paste(etiq,'-',perc,'%')

      pieC <- pie3D(perc,border = 'black',shade=0.6,theta = 1,radius=0.8,labelcex =
0.6,labelpos=NULL,
                    col=terrain.colors(5))
      pie3D.labels(pieC,radius=1,labels=labs,labelcol=par("fg"),labelcex=0.6,minsep=0.9)
    }

  })

  output$p6Sta6 <- renderPlot(res=100,{

    df <- datos$GDC

    if (!is.null(df)){

      frecs <- as.array(table(df$tissue.definition))
      frecs <- frecs[frecs>0]
      total <- sum(frecs)
      etiq <- names(frecs)
      perc <- as.array(round((frecs/total)*100,digits = 2))
      labs <- paste(names(perc),perc,'%')
      pieC <- pie3D(perc,border = 'black',shade=0.6,theta = 1,radius=0.8,labelcex =
0.6,labelpos=NULL,
                    col=terrain.colors(5))
      pie3D.labels(pieC,radius=1,labels=labs,labelcol=par("fg"),labelcex=0.6,minsep=0.9)
    }
  })

  output$p6Sta7 <- renderPlot(res=100,{

    try({
```

```
      library(GenomicDataCommons)
      datos <- cases() %>% facet('project.project_id') %>% aggregations() %>% as.data.frame()
      detach(package:GenomicDataCommons)
      colnames(datos) <- c('Proyecto','Casos')
      p<-ggplot(datos, aes(x=Proyecto, y=Casos,fill=Casos)) + geom_bar(stat="identity")+
theme_minimal() +
        coord_flip() + scale_fill_gradient(low='Green', high='red') + theme(axis.text.y =
element_text(size=7,face='bold',hjust = 0))
      p
    },silent=T)
  })

  output$p6Sta8 <- renderPlot(res=100,{

    try({

      library(GenomicDataCommons)
      datos <- cases() %>% facet('summary.data_categories.data_category') %>% aggregations() %>
% as.data.frame()
      detach(package:GenomicDataCommons)
      colnames(datos) <- c('Tipo','Casos')
      p<-ggplot(datos, aes(x=Tipo, y=Casos,fill=Casos)) + geom_bar(stat="identity")+
theme_minimal() +
        scale_fill_gradient(low='Green', high='red') + theme(axis.text.x =
element_text(size=10,face='bold',hjust = 1,angle=45))
      p
    },silent=T)
  })
```

```
#--------------------------------------------------------------------------------
# tables.R
#--------------------------------------------------------------------------------


# TABLAS REACTIVAS Y FUNCIONES RELACIONADAS

#-------------------------------------------------------------------------------------

  output$d1Stud <- DT::renderDataTable({    #(options = list(stateSave = TRUE),{


datatable(selection='multiple',escape=F,style='bootstrap',rownames=F,options=list(dom='ftp',aut
oWidth=T,stateSave=T,columnDefs=list(list(width='40%',targets=1)),
      server=T,pageLength=6,scrollX = F,scrollY=F),data=fStu()[,1:7]) # ¡¡¡ cambiar a 1:8 o 1:7
sin flags !!!

  })

#-------------------------------------------------------------------------------------

  proxySel <- dataTableProxy('d1Stud', session = shiny::getDefaultReactiveDomain(),
                        deferUntilFlush = T)

  # Funciones genéricas para manejo de proxy (devuelven los índices)

  seleccionadosGSE <- function(){

    if (is.null(input$d1Stud_rows_selected))
      return(0)
    return(as.vector(input$d1Stud_rows_selected))
  }

  todosGSE <- function(){

    return(as.vector(input$d1Stud_rows_all))
  }

#-------------------------------------------------------------------------------------

  # Desactivación de los controles de selección

  inhabConSel <- function(){
```

```
  shinyjs::disable('i1TumT')
  shinyjs::disable('i1Gend')
  shinyjs::disable('i1Race')
  shinyjs::disable('i1Ethn')
  shinyjs::disable('b1Selt')
  shinyjs::disable('b1Desc')
  shinyjs::disable('b1Limp')
  shinyjs::disable('b1Todo')
  shinyjs::disable('i1DisT')
  shinyjs::disable('b1Lanz')
  shinyjs::disable('s1Smps')
  shinyjs::disable('i1Fech')
  shinyjs::disable('i1Type')
  shinyjs::disable('i1Reps')
  shinyjs::disable('i1Pfms')
  shinyjs::disable('c1Lega')
  shinyjs::disable('i1Dist')
  shinyjs::disable('i1Orga')
  shinyjs::disable('l1Pfms')
  shinyjs::disable('l1Type')
  shinyjs::disable('l1ClPS')
  shinyjs::disable('l1SmTy')
}

# Activación de los controles de selección

actConSel <- function(){

  shinyjs::enable('i1TumT')
  shinyjs::enable('i1Gend')
  shinyjs::enable('i1Race')
  shinyjs::enable('i1Ethn')
  shinyjs::enable('b1Selt')
  shinyjs::enable('b1Desc')
  shinyjs::enable('b1Limp')
  shinyjs::enable('b1Todo')
  shinyjs::enable('i1DisT')
  shinyjs::enable('b1Lanz')
  shinyjs::enable('s1Smps')
  shinyjs::enable('i1Fech')
  shinyjs::enable('i1Type')
  shinyjs::enable('i1Reps')
  shinyjs::enable('i1Pfms')
  shinyjs::enable('c1Lega')
  shinyjs::enable('i1Dist')
  shinyjs::enable('i1Orga')
  shinyjs::enable('l1Pfms')
  shinyjs::enable('l1Type')
  shinyjs::enable('l1ClPS')
  shinyjs::enable('l1SmTy')
}

# Desactivación de los botones de la pantalla de selección de muestras

inhabConSmp <- function(){

  shinyjs::disable('b2Lim1')
  shinyjs::disable('b2Lim2')
  shinyjs::disable('b2Lim3')
  shinyjs::disable('b2Lim4')
  shinyjs::disable('b2Lim5')
  shinyjs::disable('b2Tod1')
  shinyjs::disable('b2Tod2')
  shinyjs::disable('b2Tod3')
  shinyjs::disable('b2Tod4')
  shinyjs::disable('b2Tod5')
  shinyjs::disable('b2Sele')

}

# Activación de botones de selección de muestras

actConSmp <- function(){
```

```
    shinyjs::enable('b2Lim1')
    shinyjs::enable('b2Lim2')
    shinyjs::enable('b2Lim3')
    shinyjs::enable('b2Lim4')
    shinyjs::enable('b2Lim5')
    shinyjs::enable('b2Tod1')
    shinyjs::enable('b2Tod2')
    shinyjs::enable('b2Tod3')
    shinyjs::enable('b2Tod4')
    shinyjs::enable('b2Tod5')
    shinyjs::enable('b2Sele')

  }

  # Desactivación de los botones de la pantalla de análisis
  inhabConAna <- function(){

    shinyjs::disable('b3RawD')
    shinyjs::disable('b3Beta')
    shinyjs::disable('b3Desc')
    shinyjs::disable('b3Camp')
    shinyjs::disable('b3DMRA')
    shinyjs::disable('b3Filt')
    shinyjs::disable('b3Rest')
    shinyjs::disable('b3Des2')
    shinyjs::disable('b3Impt')

  }

  # Activación de botones de la pantalla de análisis

  actConAna <- function(){

    shinyjs::enable('b3RawD')
    shinyjs::enable('b3Beta')
    shinyjs::enable('b3Desc')
    shinyjs::enable('b3Camp')
    shinyjs::enable('b3DMRA')
    shinyjs::enable('b3Filt')
    shinyjs::enable('b3Rest')
    shinyjs::enable('b3Des2')
    shinyjs::enable('b3Impt')

  }

  # Reinicio de los datos reactivos

  resetData <- function(){

    datos$gGSE <- NULL
    datos$GSEsel <- NULL
    datos$GSMsel <- NULL
    datos$importa <- NULL
    datos$infoGSEs <- NULL
    datos$infoGSMs <- NULL
    datos$GDC <- NULL
    datos$methContainer <- NULL
    datos$anotaciones <- NULL
    datosDMR$DMRs <- NULL
    datos$mensajes <- NULL
    datos$notificaciones <- NULL

  }

#-------------------------------------------------------------------------------

  # Consulta a los diferentes repositorios, devuelve data.frame a tabla dinámica

  # MEJORA: si sólo se busca en ArrayExpress, no se mostrarían resultados redundantes con GEO
(sólo los gestionados por ArrayExpress aunque también están almacenados los de GEO)

  # stopifnot function

  fStu <- function(){
```

```
keyWds <- parseSep(control$gKeywd)
operador <- control$gSchop
iRepo <- isolate({input$i1Reps})
iPlat <- input$i1Pfms
iSmps <- input$s1Smps
iFech <- input$i1Fech
iTipo <- isolate({input$i1Type})
iClas <- isolate({input$i1DisT})
iLega <- isolate({input$c1Lega})
iOrga <- input$i1Orga
iOrPla <- control$gOrPla
inhabConSel()
resetData()
GSEs <- NULL
GSEs2 <- NULL
# Filtro GEO
if (iRepo=='GEO'){

  tryCatch({
    GSEs <- filtroGEO(keyWds,operador,iPlat,iSmps,iFech,iTipo,iOrga)

  },error=function(GSEs){
      GSEs <- NULL
      ventanaError('Problemas de acceso a GEO')
  })
}


# Filtro ArrayExpress
if (iRepo=='ArrayExpress'){

  tryCatch({
    GSEs <- filtroAE(keyWds,operador,iSmps,iFech,iTipo,iOrga,iPlat)
  },error = function(GSEs) {
      GSEs <- NULL
      ventanaError('Problemas de acceso a ArrayExpress')
  })
}

# Filtro GDC
if (iRepo=='GDC'){
  OK <- F
  updateSelectInput(session,'i1Orga',selected='Homo sapiens')
  try({
    OK <- TCGAbiolinks::isServeOK()
  },silent=T)

  if (OK){
    tryCatch({
      GSEs <- filtroGDC(iSmps,iFech,iTipo,iClas,iLega)
      },error=function(GSEs){
      GSEs <- NULL
      ventanaError('Error en la consulta GDC')
    })
  }
  else{
    GSEs <- NULL
    ventanaError('Problemas de acceso a GDC')
  }

}

if (iRepo=='Todos')
{
  GSEs1 <- NULL
  GSEs2 <- NULL
  GSEs3 <- NULL
  errorGEO <- T
  errorAE <- T
  errorGDC <- F
  msjeError <- 'Problemas de acceso a '
  # Check acceso a GEO
  try({
```

```
        GSEs1 <- filtroGEO(keyWds,operador,iPlat,iSmps,iFech,iTipo,iOrga)
        errorGEO <- F
      },silent=T)

      # Check acceso a ArrayExpress
      try({
        GSEs2 <- filtroAE(keyWds,operador,iSmps,iFech,iTipo,iOrga,iPlat)
        errorAE <- F
      },silent=F)

      # Check acceso a GDC
      if ((grepl('Homo sapiens',iOrga,ignore.case = T)&&iOrPla==F)||(iOrPla==T)){

        OK <- F
        try({
          OK <- TCGAbiolinks::isServeOK()
          OK <- T
        },silent=T)

        if (OK)
          try({
            errorGDC <- T
            GSEs3 <- filtroGDC(iSmps,iFech,iTipo,input$i1DisT,iLega)
            errorGDC <- F
          },silent=T)
      }

      # Errores
      if (errorGEO){
        msjeError <- paste(msjeError,'GEO')
        GSEs1 <- NULL
      }
      if (errorAE){
        msjeError <- paste(msjeError,'ArrayExpress')
        GSEs2 <- NULL
      }
      if (errorGDC){
        msjeError <- paste(msjeError,'GDC')
        GSEs3 <- NULL
      }

      # Fusión de resultados
      GSEs <- rbind(GSEs1,GSEs2)
      GSEs <- rbind(GSEs,GSEs3)


      if ((errorGEO==T)||(errorAE==T)||(errorGDC==T))
        ventanaError(msjeError)
    }

    # En caso de no haber errores, los filtros podrían devolver dataframes vacíos
    try({if (dim(GSEs)[1]==0) GSEs <- NULL},silent=T)
    datos$gGSE <- GSEs

    actConSel()

    return(GSEs)
  }


#-------------------------------------------------------------------------------

  output$d2Smp1 <- DT::renderDataTable({    #(options = list(stateSave = TRUE),{
    datatable(selection='multiple',escape=F,style='bootstrap',rownames=F,options=list(stateSave
= TRUE,dom='tp',autoWidth=T,stateSave=T,
      server=F,pageLength=6,scrollX = F,scrollY=F),class='table-bordered',data=fSmpls(1))
  })

  output$d2Smp2 <- DT::renderDataTable({    #(options = list(stateSave = TRUE),{


datatable(selection='multiple',escape=F,style='bootstrap',rownames=F,options=list(dom='tp',auto
Width=T,stateSave=T,
      server=T,pageLength=6,scrollX = F,scrollY=F),class='table-bordered',data=fSmpls(2))
```

```
  })

  output$d2Smp3 <- DT::renderDataTable({    #(options = list(stateSave = TRUE),{

datatable(selection='multiple',escape=F,style='bootstrap',rownames=F,options=list(dom='tp',auto
Width=T,stateSave=T,
      server=T,pageLength=6,scrollX = F,scrollY=F),class='table-bordered',data=fSmpls(3))

  })

  output$d2Smp4 <- DT::renderDataTable({    #(options = list(stateSave = TRUE),{

datatable(selection='multiple',escape=F,style='bootstrap',rownames=F,options=list(dom='tp',auto
Width=T,stateSave=T,
      server=T,pageLength=6,scrollX = F,scrollY=F),class='table-bordered',data=fSmpls(4))

  })

  output$d2Smp5 <- DT::renderDataTable({    #(options = list(stateSave = TRUE),{

datatable(selection='multiple',escape=F,style='bootstrap',rownames=F,options=list(dom='tp',auto
Width=T,stateSave=T,
      server=T,pageLength=6,scrollX = F,scrollY=F),class='table-bordered',data=fSmpls(5))

  })

#-------------------------------------------------------------------------------

  proxySmp1 <- dataTableProxy('d2Smp1', session = shiny::getDefaultReactiveDomain(),
                              deferUntilFlush = T)

  proxySmp2 <- dataTableProxy('d2Smp2', session = shiny::getDefaultReactiveDomain(),
                              deferUntilFlush = T)

  proxySmp3 <- dataTableProxy('d2Smp3', session = shiny::getDefaultReactiveDomain(),
                              deferUntilFlush = T)

  proxySmp4 <- dataTableProxy('d2Smp4', session = shiny::getDefaultReactiveDomain(),
                              deferUntilFlush = T)

  proxySmp5 <- dataTableProxy('d2Smp5', session = shiny::getDefaultReactiveDomain(),
                              deferUntilFlush = T)

  # Funciones genéricas para manejo de proxy

  todosSmp1 <- function(){

    return(as.vector(input$d2Smp1_rows_all))
  }

  todosSmp2 <- function(){

    return(as.vector(input$d2Smp2_rows_all))
  }

  todosSmp3 <- function(){

    return(as.vector(input$d2Smp3_rows_all))
  }

  todosSmp4 <- function(){

    return(as.vector(input$d2Smp4_rows_all))
  }

  todosSmp5 <- function(){

    return(as.vector(input$d2Smp5_rows_all))
  }

#-------------------------------------------------------------------------------
```

```
  fSmpls <- function(indice){

    muestras <- datos$infoGSMs[[indice]]
    return(muestras)

  }

#-------------------------------------------------------------------------------

  output$d3Mues <- DT::renderDataTable({      #(options = list(stateSave = TRUE),{

datatable(selection='none',escape=F,style='bootstrap',rownames=F,options=list(dom='tp',autoWidt
h=T,stateSave=T,

server=T,pageLength=3,scrollX = F,scrollY=F),class='table-bordered',data=datos$GSMsel[,-9])

  })


  #-------------------------------------------------------------------------------

  output$d3Proc <- DT::renderDataTable({    #(options = list(stateSave = TRUE),{

datatable(selection='none',escape=F,style='bootstrap',rownames=F,options=list(dom='tp',width='1
00%',stateSave=T,

server=T,pageLength=10,scrollX = T,scrollY=F),class='table-bordered',data=datos$procesados)

  })

  #-------------------------------------------------------------------------------

  output$d3Impo <- DT::renderDataTable({    #(options = list(stateSave = TRUE),{
    datatable(selection = list(target =
'column'),escape=F,style='bootstrap',rownames=F,options=list(dom='t',pageLength=5,scrollX =
T,scrollY=T),class='table-bordered',data=importa())

  })

  #-------------------------------------------------------------------------------

  importa <- reactive({

    inFile <- input$w3file

    if (is.null(inFile))
      return(NULL)
    else
    {
      tryCatch({

        read.table(inFile$datapath, header = input$c3Head,sep=input$r3Sepa,stringsAsFactors =
F,dec='.')

        },error=function(e){

        },finally={
          read.table(inFile$datapath,header = input$c3Head,sep=input$r3Sepa,stringsAsFactors =
F,dec='.',fileEncoding='UTF-16LE')

      })

    }

  })


  #-------------------------------------------------------------------------------

  output$d3DMRs <- DT::renderDataTable({    #(options = list(stateSave = TRUE),{
```

```
datatable(selection='none',escape=F,style='bootstrap',rownames=F,options=list(dom='tp',width='1
00%',stateSave=T,

server=T,pageLength=10,scrollX = T,scrollY=F),class='table-bordered',data=datosDMR$DMRs)

  })
```